目　录

第 1 章　MatLab 简介及使用指南

MatLab 功能很强大，能够进行科学计算和大量的工程运算。并且，通常这些运算人工难以实现的，即使能够实现，其工作量也是非常可观的，其编程也极其复杂。而 MatLab 工具使这些计算变得简单而且准确有效，其产生的工作进程和效率是用通常的编程方法所无法比拟的。MatLab 不单功能强大，而且简单易用，只要有一点 Windows 操作的经验，甚至不需要任何基础知识就可以在短时间内快速掌握其主要内容和基本操作，并且运用其解决大量手工难以完成的工作，成为一个地地道道的计算高手。

1.1　安装及其版本介绍

MatLab 的最初版本是由 FORTRAN 语言编写的矩阵分析软件，用来提供与 LINPACK 和 EISPACK 矩阵软件包接口。

MatLab 的最新版本占内存空间非常大，如果将所有的工具箱及帮助文件都安装，则需要几百兆磁盘空间，即使有选择地最小化安装也大约需要 70 兆的磁盘空间。推荐使用 Win95 以上操作系统及 32M 以上内存，因为 MatLab 进行图形处理时需要很大的内存空间。利用 MatLab 的安装向导来安装 MatLab 是非常方便的，下面介绍 MatLab 的安装过程。

(1) 将存有 MatLab 的光盘放入光驱，运行 setup.exe 命令，就会出现以下画面，如图 1.1 所示。此对话框是 MatLab 的欢迎信息，一般安装程序都会出现此类窗口，提示

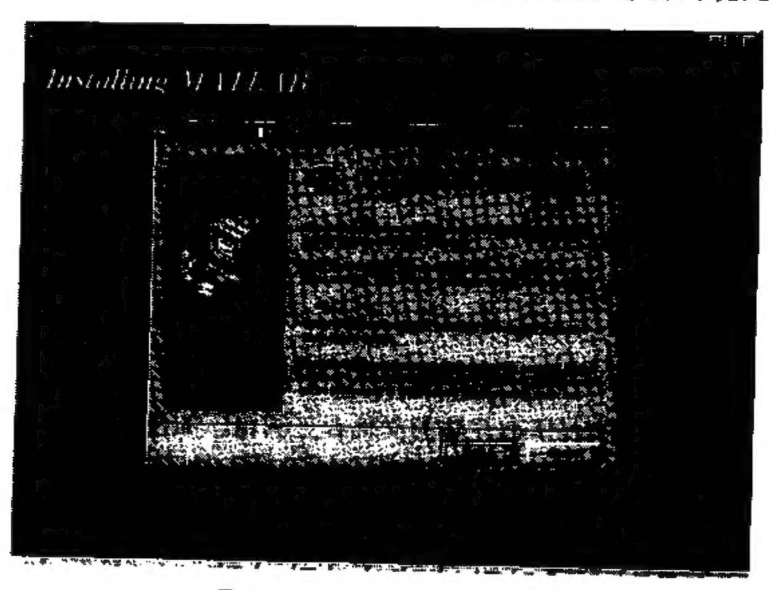


图 1.1　MatLab 安装欢迎界面

版权等信息。单击 Next 按钮即可以进入第(2)步操作，出现 Customer Information 对话框如图 1.2 所示。

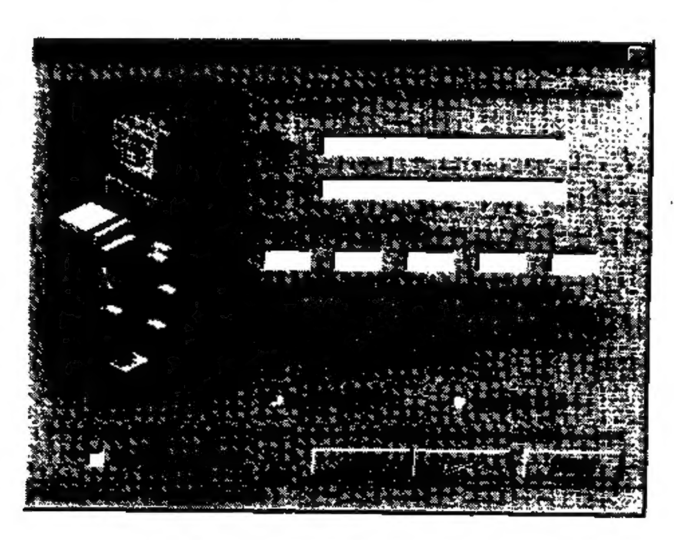


图 1.2 用户信息及权限验证对话框

(2) 在该对话框中要求输入用户姓名和公司名称，并输入注册号以进行验证，如果输入的注册号不正确，则 MatLab 会自动推出安装向导，单击 Next 按钮，进入第(3)步操作，出现的对话框如图 1.3 所示。

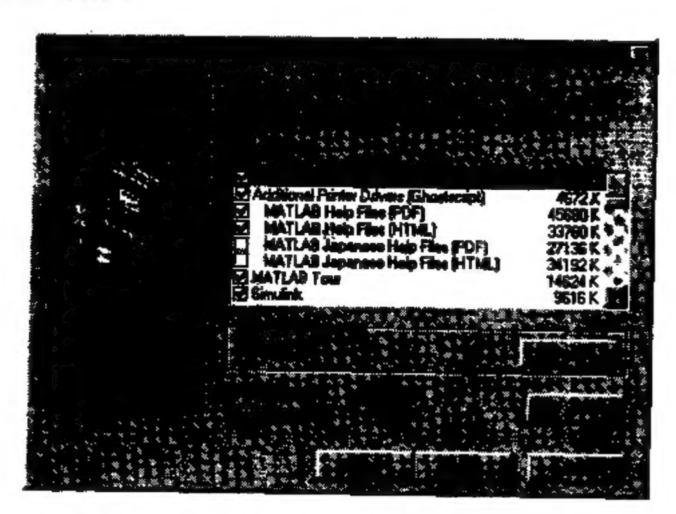


图 1.3 选择要安装的文件和安装路径

(3) 在该对话框中可以选择需要安装的文件或工具箱，如果某项方框内有“√”，则表示此项需要安装。MatLab 的完全安装需要很大的磁盘空间，但有许多工具箱及一些帮助文件一般用不到，建议在安装时应该详细了解所要用到的功能和各个工具箱的作用，将用不到的选项删除以尽量节省磁盘空间。通过单击 Browse 按钮可以选择安装路径，在选择路径时应该考虑所需的磁盘空间和现有的磁盘空间，如果没有足够的磁盘空间，则安装不会正常进行。单击 Next 按钮进入第(4)步操作，出现的窗口如图 1.4 所示。

(4) 主要是将文件拷贝到磁盘中，屏幕中间的进度条反映了文件拷贝所完成的进度。

在拷贝文件完成后，MatLab的安装程序会提示安装完成，最后单击Finish按钮即完成了安装，经过重新启动计算机，就可以顺利使用了。

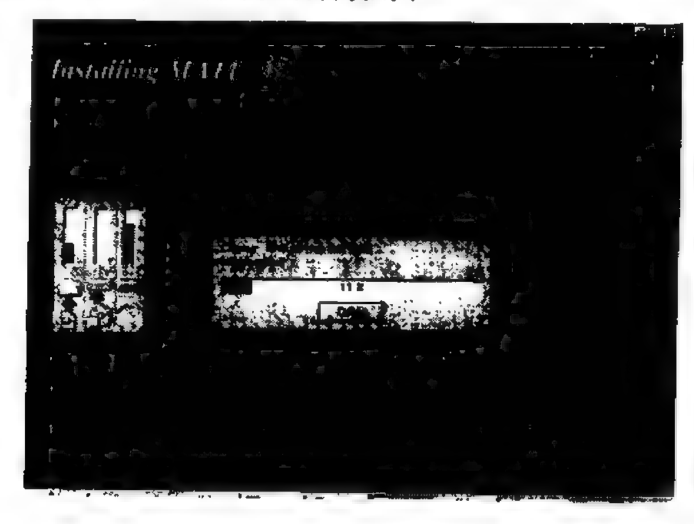


图1.4 拷贝文件到硬盘中

1.2 MatLab的界面与基本操作入门

本节介绍如何以不同方式进入MatLab。为了能够更快地理解和掌握MatLab执行命令的方式，还将介绍一个简单的例子，通过这个例子可以很快体会到用MatLab对矩阵进行计算和操作确实方便快捷。

1.2.1 进入MatLab操作环境

在Win95/98环境下，单击【开始】|【程序】命令，找到MatLab的图标，双击即可打开MatLab并进入其操作环境。如图1.5所示。



图1.5 从【开始】菜单进入MatLab的操作环境

此外，到 MatLab 目录下，如在本地目录 F:\MATH SOFTWARES\MatLab\bin 找到 MatLab 的图标，双击即可打开，如图 1.6 所示。

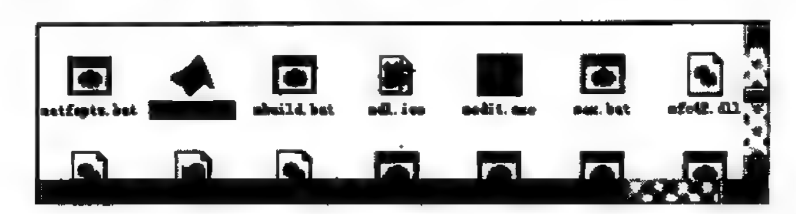


图 1.6 双击 MatLab 应用程序图标进入 MatLab 环境

然后，即可进入 MatLab 操作环境，如图 1.7 所示。

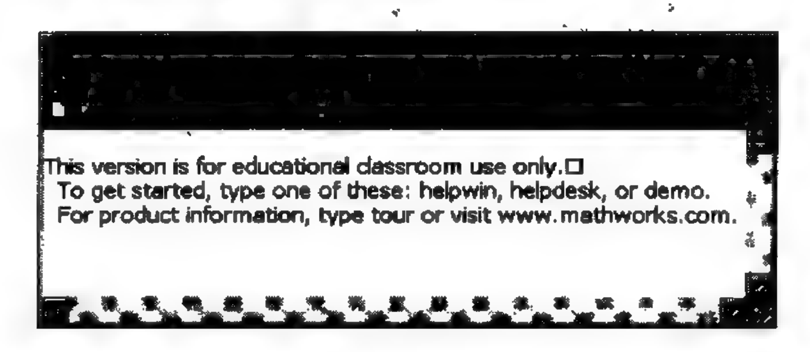


图 1.7 MatLab 的命令窗口

最上面一栏为 MatLab 的菜单栏，单击或按组合键 Alt+Key(Key 为菜单栏上有下划线的字母)即可打开相应的菜单。如按组合键 Alt+F 即可打开 File 菜单栏从而进行与文件有关的各种操作，按组合键 Alt+E 即可打开 Edit 菜单进行编辑。将在 1.3 节详细介绍各个菜单栏的使用方法和作用。

在菜单栏的下面是快捷工具栏，其上都是经常用到的命令，掌握和使用它们通常能使操作更简便快捷，从而避免打开冗长的菜单栏，在 1.3 节中将逐一介绍这些命令的功能，以便快速掌握其用法。

下面一块空白区域是 MatLab 的工作区，在此可以输入命令并可立即得到执行。

1.2.2 一个简单的例子

暂不考虑 MatLab 的具体用法和语法知识，为了对 MatLab 的工作方式有一个感性的认识，现举一个简单的矩阵的例子，因为 MatLab 的重要功能就是矩阵的运算。在工作区中输入以下命令：

```
A = [16 3 2 13; 5 10 11 8;  9 6 7 12;  4 15 14 1]
```

这是一个建立矩阵的命令，A 矩阵是 3×3 的方阵，矩阵各行间用“；”隔开，每个矩阵都用“[]”括起，输入后按回车键即可。得到返回结果如下：

```
A =
     16     3     2     13
```

```
        5    10    11     8
        9     6     7    12
        4    15    14     1
```

表明 A 矩阵已经建立并存入计算机的内存中，可以随时使用它，只要调用它，它就会出现。如在工作区输入 A，按回车键，在工作区内即刻显示 A 的内容，同上面显示结果一致。

刚才所建立的矩阵是首先出现在德国艺术家及业余数学家 Albrecht Dürer 的雕刻作品中的魔方阵。通过具体计算会发现这个魔方有着某种值得探讨的特性，如果将纵向的或横向的或沿任一对角线的 4 个数相加，会得到同一个数 34。下面用 MatLab 强大的矩阵计算功能来验证一下。

在工作区中输入 sum(A)便会得到：

```
ans =
    34    34    34    34
```

其中，A 便是上面刚刚建立的魔方阵，函数 sum 是 MatLab 的内置函数，其作用是对矩阵的个列求和并返回个列和组成的行向量。由此结果可知，A 矩阵的个列和相等并且都等于 34。ans(answer 的缩写)是 MatLab 的一个变量，它与一般的变量没什么区别，只是 MatLab 会自动将没指定返回变量的结果赋值给 ans。

如果输入 b=sum(A)则将会得到：

```
b =
    34    34    34  34
```

因为将结果赋给了变量 b，因而不会出现 ans。

我们已经得到了各列的值，下面将计算各行的值。可以通过运算符“'”来实现矩阵的转置，然后将所得结果转置，即可得到各行的和。运算符“'”将会使矩阵绕其主对角线翻转，或将一个行(列)向量转为一个列(行)向量。输入：

```
A'
```

便会得到：

```
ans =
        16       5        9       4
        3        10       6       15
        2        11       7       14
        13       8        12      1
```

可以看出这正是 A 矩阵的转置。将它赋给变量 c：

```
c=A'
```

然后计算 c 矩阵的个列的和，即为 A 矩阵的各行的和。输入：

```
d=sum(c)
```

便得到：

```
d =
```

```
    34    34    34    34
```

然后将其转置，即可得到 A 各行的和：

```
e=d'
e =
    34
    34
    34
    34
```

到此为止已经计算出了 A 的各行和各列的和，都等于 34，现在考察变量 ans 的值，输入 ans 得到：

```
ans =
     16        5         9         4
     3         10        6         15
     2         11        7         14
     13        8         12        1
```

说明 ans 的值并不等于 e 的值，这是因为以后的计算中将结果赋值给了各个变量。下面再考察一下两对角线各原色的和。用函数 diag()取出矩阵的主对角线的元素，形成一个列向量，计算过程如下：

```
diag(A)
ans =
    16
    10
     7
     1
sum(diag(A))
ans = 34
```

可知 A 矩阵主对角线元素的和也等于 34。另一条副对角线在数学上的地位不重要，因而 MatLab 中并没有现成的能够直接取出其副对角线的函数，但可以利用现成的其他函数来完成这一步。

下面来介绍一个在图形变换中很重要的函数 fliplr，它将矩阵左右翻转。

例 1　e=fliplr(A)

```
e =
    13     2     3    16
     8    11    10     5
    12     7     6     9
     1    14    15     4
```

然后依照上面，进行下面的操作：

```
sum(diag(e))
ans =
    34
```

通过简单的操作，完成了这个魔方阵的验证，相信用户现在应该对 MatLab 的操作有了一个大概的了解。

1.3　具体操作介绍

从图 1.7 中可以看出，MatLab 的基本界面是典型的图形窗口界面，并且非常简洁，并没有太多的菜单和按钮。本节主要介绍部分菜单和按钮的操作，这部分操作主要是对 MatLab 的工作空间进行管理及文本编辑的一些命令，还有一些 MatLab 所特有的命令，如管理变量的命令。MatLab 的计算功能并没有在这一部分中体现出来，如果对 Windows 操作界面非常熟悉，那么学习这一节将比较容易，因为其中的绝大部分菜单命令都是典型的 Windows 菜单命令。MatLab 的菜单包括 File，Edit，Window，Help 4 组，这在很多 Windows 的应用程序中是非常普遍的，而且各组所包含的功能也大致相同。File 菜单主要包含对文件进行操作及对工作空间进行设置的一些命令；Edit 菜单包含文本编辑的一些命令；Window 菜单包含对窗口操作的命令；Help 菜单包含与帮助有关的命令。下面对各菜单按功能进行逐一介绍。

1.3.1　文件操作

对文件的操作是指打开、关闭或保存文件等的操作，而这些操作在大多数的应用软件中几乎都是通用的，在此就不再赘述。而 MatLab 中有一个新建并编辑 M 文件的操作，会打开 MatLab 自带的一个文本编辑器，提供编写 M 文件的友好界面，其使用也比较简单，可以很容易地掌握。

单击 File|New|M-file 命令，运行时建立新的 M 文件，同时打开 MatLab 提供的 M 文件编辑器和调试窗口。M 文件为普通的文本格式，因而可以用任何编辑器来编辑，包括 Dos 下的 Edit 编辑器。事实上，在工作区中输入“！edit”即可进入 DOS 状态并进入 Edit 编辑器进行 M 文件的编辑，而不加“！”则会直接调用 MatLab 自带的 M 编辑器，等同于 New|M-file 命令。界面如图 1.8 所示。

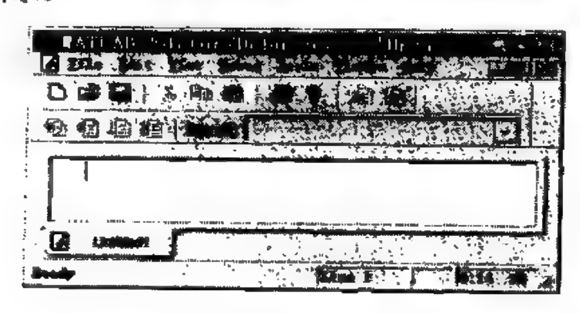


图 1.8　M 函数和 M 文件编辑器

1.3.2　工作空间操作

工作空间简单说就是运行状态下的 MatLab 所提供的开发环境，而开发环境中的主要

内容就是所建立的变量，因而对工作空间的操作就主要是针对这些变量进行操作。

1. Load workspace

载入工作空间文件，通过对工作空间文件的调用，可以恢复上次 MatLab 的环境状态，包括所用的变量。将会在 1.4 节详细讨论。

2. Save workspace As

将当前的工作空间更名保存，便于以后用 load workspace 调用此文件恢复当前的状态。这条命令很有用，当因有特殊的情况而不能继续当前的计算时，完全可以保存当前的状态到文件中，当需要时，再恢复此状态继续计算。

3. Show workspace

显示当前工作空间中的信息，提供对矩阵变量、内置对象、向量、字符串变量的图形方式的浏览，它实际上是 whos 命令的图形方式的显示，如图 1.9 所示。

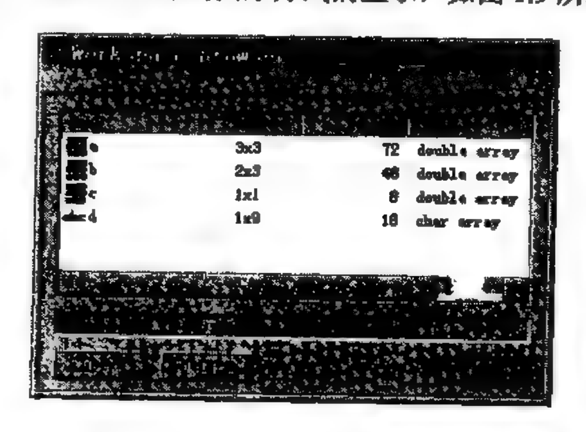


图 1.9 MatLab 工作空间中的变量浏览器

比较此命令与 whos 命令显示方式上的不同，介绍如下：

```
whos
  Name       Size         Bytes  Class

  a          3x3             72  double array
  b          1x3             24  double array
  c          3x1             24  double array
  d          3x3             72  double array
  e          3x3             72  double array
  f          1x1              8  double array
Grand total is 34 elements using 272 bytes
```

1.3.3 路径操作

MatLab 中，路径实际上是作为一个环境参数出现的。在 MatLab 对函数或文件等进行

搜索时，都是在其搜索路径下进行的。可以通过以下菜单选项对路径进行增加或删除。

单击 File|Set path 命令，打开编辑路径的对话框，进行通过 path 命令所作的各种操作，如增加或删除路径，并进行路径的浏览和管理界面，如图 1.10 所示。

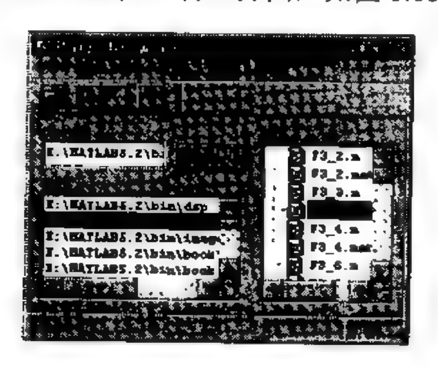


图 1.10 MatLab 路径管理器

在 Current 文本框中显示了 MatLab 的当前路径，通过单击 Browse 按钮可以进行修改，操作同一般的路径选择对话框。

Path 列表框显示了 MatLab 的所有路径。而右边 Files in My 列表框则显示了在 Path 列表框中所选择的路径下的所有文件。

下面介绍 4 个工具栏按钮的功能：

当选中右边 Files in My 列表框中的一个 M 文件时，单击它可以打开并编辑该文件。

通过直接输入或浏览方式将一个路径加入到 MatLab 的路径列表中，如图 1.11 所示。

当选中一个可以删除的路径时，便可以通过按此按钮来删除路径。

该按钮表示调用当前对话框的帮助文件或“关于”信息，在此是“关于”信息之意。

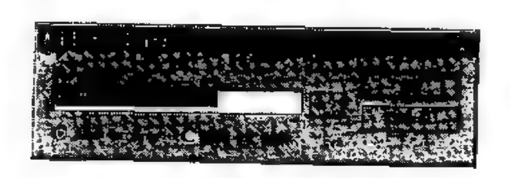


图 1.11 向搜索路径中增加新的路径

1.3.4 打印操作

对数据进行计算或绘制出图后，要对结果进行打印输出。打印是极为普通的一种输出，MatLab 中有打印和打印设置的命令，与其他应用程序的打印命令基本相同。在早期的 MatLab 版本中，不能直接对图形进行打印，并且在打印图形前必须先运行屏幕硬拷贝程序，为了不致过多地占用内存，通常通过中间文件进行打印。高版本的 MatLab 也有这方面的功能，在 MatLab 工作命令窗口中输入 Print filename 可以打印图形。

1. File|Print Setup

对打印进行设置，可以设置打印机的名称、纸张的大小、来源、打印的方向、打印的字体和格式等。如图 1.12 所示。

2. File|Print

进行打印，在此也可以设置打印机及打印范围和份数，并且可以通过【打印到文件】选项将设置保存到.prn 文件中。如果没有打印机，可以先将内容保存到一个文件中，然后到有打印机的地方打印输出。

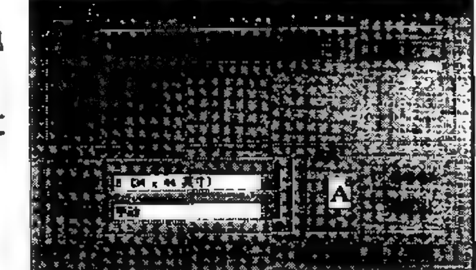


图 1.12 打印设置

此外，在 File 菜单的最下面，列出了最近使用过的 4 个 M 文件，可以单击这些选项直接打开这些文件，从而避免了冗长的目录搜索。

1.3.5 MatLab 的参数设置

此项实际上是 File 菜单中的 Preferences 的内容，为了对 MatLab 的菜单功能有一个整体的了解，将其单独列出。

单击 File|Preferences 命令显示其参数设置的对话框，共包括 General，Copying Options，Command Window Font 3 个选项卡，下面分别介绍。

1. General 选项卡

如图 1.13 所示，在 General 选项卡中可以设置工作区中数字显示的方式。其中各选项的意义如表 1.1 所示。

表 1.1 数字显示格式

选 项	含 义
short	短格式
Long	长格式
Hex	十六进制格式
Bank	银行格式
Plus	紧密格式
Short E	短格式 E 方式
Long E	长格式 E 方式
Short G	短小数格式
Long G	长小数格式
Rational	有理数格式



图 1.13　设置数据显示格式的参数选项卡

对于上面格式的显示方式，将在 1.4 节中以命令方式介绍，每一种格式都可以通过命令方式来实现。

Editor Preference 框架中的选项是用 MatLab 内嵌的文本编辑器来编辑 M 文件，还是用其他自定义的编辑器，通过 Browse 按钮可以选择编辑器的路径。在 Help Directory 框架中的选项则指出了 MatLab 的帮助文件的路径。下面的开关按钮中，Echo On 复选框指明是否要回显，与 DOS 的批处理文件中的 echo on/off 命令作用一样。Show Toolbar 开关决定是否显示工具栏。

Enable Graphical Debugging 复选框允许图形方式的调试。Always Reload Network Directory 复选框如果计算机已经联网，在前面路径中，可以将另一台计算机上的共享路径加入 MatLab 的路径中，用这个参数可以在每次打开 MatLab 时自动恢复与另一台计算机的网络连接。

2. Copying Options 选项卡

如图 1.14 所示，Copying Options 选项中各项的含义介绍如下。

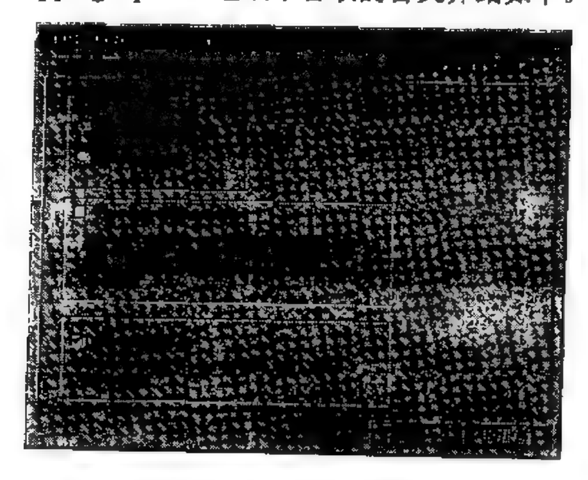


图 1.14　Copying Options 选项卡

Windows Metafile　该单选按钮以 window 图元文件的格式将当前图像存入剪贴板。

Windows Bitmap　该单选按钮以 window 位图文件的格式存入剪贴板。

Honor figure size properties　该复选框为图像大小的属性，如果选中此项，在复制图像时可以单击 File|PaperPosition 命令来指定要拷贝的图形的尺寸大小，如未选中它，则将所有显示在屏幕上的图形都拷入剪贴板。

3. Command Window Font 选项卡

如图 1.15 所示，在此选项卡的选项中，可以设置 MatLab 工作区中显示的字体的风格和颜色。在 Font 框架中可以设置字体名称，如隶书、幼圆等，Style 文本框设置字体风格，如加粗等；Size 文本框设置字体显示的大小，Background Color 设置工作区空白区的颜色，Color 下拉列表框用来设置字体的颜色。

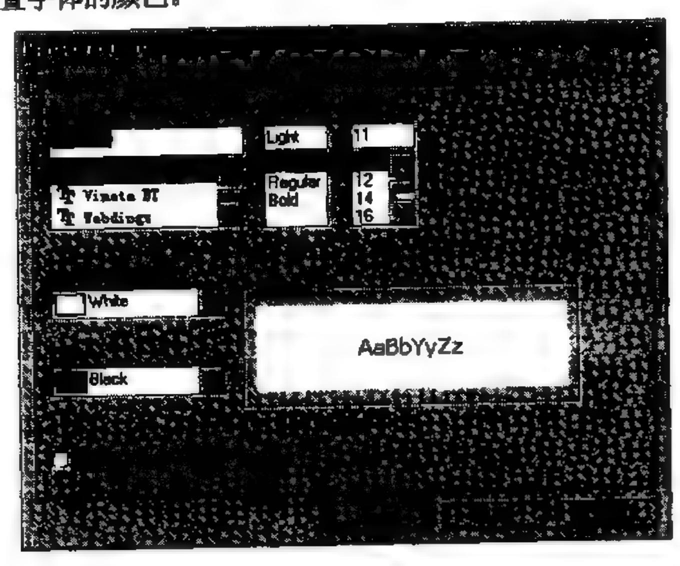


图 1.15　Command Window Font 选项卡

设置完成后可以通过 Sample 区域对设置进行效果预览，如果满意，可以通过单击应用使刚才的设置直接生效而不退出此设置环境，继续进行其他的设置。如果单击【确定】按钮，则当前设置生效并退出，返回到 MatLab 工作区，单击【取消】按钮会取消此次设置并返回到工作区，设置不生效。其他对话框中的类似的 3 个按钮作用亦同。

1.3.6　文本编辑操作

主要是对工作区中显示的文本的剪贴、拷贝、清除等操作。这是典型的文本编辑菜单，其主要功能如下：

Edit|Undo	撤销上次操作。
Edit|Cut	对选择的文本进行剪切。
Edit|Paste	将剪贴板中的内容贴入当前区域。
Edit|Clear	清除工作区。
Edit|Select All	选中工作区中所有的文字。

以上命令是文本编辑器中常用命令，在命令窗口中也可用。

1.3.7 帮助操作

由于学会使用 MatLab 提供的帮助内容非常重要，将单独在 1.5 节中详细介绍各种帮助命令的作用与用法。在此只列出各菜单的简单作用。

Help|Help window　　调用 MatLab 的帮助窗口。
Help|Help tips　　显示 MatLab 的帮助技巧。
(以上两条菜单事实上是调用同一界面，只是显示内容有所区别而已)
Help|Help desktop(HTML)　　显示 MatLab 超文本格式的帮助桌面。
Help|Examples and demos　　MatLab 的示例与显示。
Help|About MatLab　　关于 MatLab 的版本等信息。

1.3.8 工具条操作

工具条的各个按钮如图 1.16 所示。其功能和相应菜单命令如表 1.2 所示。



图 1.16 工具栏

表 1.2 工具条按钮功能简表

按钮序号	功　能	相应菜单命令
1	新建 M 文件	File|New|M-file
2	打开文件	File|Open
3	剪切	Edit|Cut
4	复制	Edit|Copy
5	粘贴	Edit|Paste
6	撤销上次操作	Edit|Undo
7	显示工作空间中变量	File|Show Workspace
8	路径管理	File|Set path
9	帮助	Help|Help Tips

1.4 通 用 命 令

通用命令是 MatLab 中应用率非常高的一组命令，这些命令可以用来管理命令和函数、管理变量和工作空间、对文件进行操作、控制窗口以及获得帮助信息和其他的信息。为了更好地管理和使用 MatLab，希望能够熟练掌握和理解这些命令。在 1.4 节中将对这些命令进行分类介绍。

1.4.1 管理命令和函数介绍

此部分的函数主要用来获得 MatLab 的函数及其搜索路径下的文件的一些信息，如函数的基本用法等，这些函数和命令对初学者来说是非常有用的。

1. 获得 MatLab 函数和 M 文件的在线帮助

MatLab 对大多数函数都有在线说明，如果在使用 MatLab 时不知道某个函数的用法，可以很方便地利用 help 命令来获得此函数的使用说明。如果只输入 help，MatLab 会列出所有最基础的帮助主题，每一条帮助主题都对应着一个 matalab 的搜索路径；如果输入 help topic，MatLab 会给出关于 topic 主题的帮助信息，“topic”可以为一个函数名、路径名、或部分路径名，以及在 MatLab 搜索路径中的一个 M 文件名。如果输入一个函数名，MatLab 输出关于这个函数的信息，如果是路径名，则会显示该路径的目录文件。目录文件就是该目录下名为 contents.m 的文件，此文件一般包括此目录下的所有命令和函数的基本信息和作用。如果此目录下没有此目录文件，则 MatLab 会自动显示此目录下的每个文件的第 1 行注释，即所谓的 H1 注释行。在输入各路径时，不必要输入路径的全部(通常这是非常费力的一件事，因为 MatLab 的路径名都非常长)，只要输入路径的最后一部分，或最后几部分(通常当在不同的目录下存在两个或两个以上的同名子目录时需这样做)，MatLab 即可定位目录。

例 2　help

```
HELP topics:
bin\myfun             -  (No table of contents file)
MatLab\bin            -  (No table of contents file)
MatLab\datafun        -  Data analysis and Fourier transforms.
MatLab\datatypes      -  Data types and structures.
MatLab\demos          -  Examples and demonstrations.
    ⋮                         ⋮
powersys\powersys     -  Power System Blockset
database\database     -  Database Toolbox.
database\dbdemos      -    Database Toolbox Demonstration Functions.
java\MatLab           -  (No table of contents file)

For more help on directory/topic, type "help topic".
```

例 3　将当前路径设置为\MatLab\toolbox\map\map

可以通过 set path 命令或直接输入 cd 'F:\MATH SOFTWARES\MatLab\toolbox\map\map'到此目录。

```
①   help map
Mapping Toolbox
  Version 1.0.1  21-Nov-1997
   What's new.
    Readme    - Up-to-date information.
```

```
  Points.
    antipode  - Compute the point on the opposite side of the globe.
    azimuth   - Azimuth between two points.
    distance  - Distance between two points.
    reckon    - Reckoning from a starting point, with an azimuth and range.
    departure - Departure of longitudes at specific latitudes.
    ⋮                         ⋮
```

②type contents.m

```
% Mapping Toolbox
% Version 1.0.1  21-Nov-1997
%
% What's new.
%   Readme    - Up-to-date information.
%
% Points.
%   antipode  - Compute the point on the opposite side of the globe.
%   azimuth   - Azimuth between two points.
%   distance  - Distance between two points.
%   reckon    - Reckoning from a starting point, with an azimuth and range.
%   departure - Departure of longitudes at specific latitudes.
    ⋮                         ⋮
```

比较上述2例可以发现，help map 所显示的内容正是目录\MatLab\toolbox\map\map 下 contents.m 文件的内容。

例4 help bin

横放圆柱

```
This is the machine-generated representation of a Handle Graphics object
```

到 bin 目录下，并没有发现 contents.m 文件。仔细打开此目录下的每个 M 文件就会发现，"横放圆柱"是 heat4_44.m 文件的第1条注释行，而下面一行是 flow_pic.m 文件的第1条注释行(以"%"开头的行)。

理解了上述命令以后，不难发现，可以写关于自编的 M 文件及工具箱的帮助信息。方法很简单，只需要建立名为 Contents.m 的文本文件，或在 M 文件中加入"%"开头的注释行即可。

如果 topic 为函数名，则 help topic 会显示函数的 M 文件中的第一块注释区中的信息。

例 5 用 edit 建立如下 M 文件：myfun(x)

```
function d=myfun(x)
%this is the example
%
%  for the help topic
%enjoy it
if x<0;d='less than o';
else if x==0; d='this is avalible'; error('Incorrect number of arguments');
```

```
else if x==1; d='one ';
   else
      d='more than one';
end

%  this end of the example
% have you caught that?
```

并保存。然后在工作区中输入：

```
help myfun
this is the example

for the help topic
    enjoy it
```

可以发现，help 只显示了 myfun 函数中第 1 块相连的注释行，而第 2 块不显示。

2. 获得帮助文件的超文本格式的说明

在 MatLab 中，关于一个函数的在线帮助信息可以用 doc 命令以超文本的方式给出，如 Doc function。如果不带参数，MatLab 会自动装载超文本格式的帮助桌面。如果带有函数名作为参数，MatLab 显示有关此函数的超文本格式的说明及应用、参数和功能等方面的信息。如果此函数被重载，即在不同的路径下存在不只一个名为 function 的函数，doc 命令会在工作区中显示所有名为 function 的重载函数的路径和定位各不同路径下的 function 的方法。

例 6 比较下面 doc 命令的 3 种用法

```
①doc
```

打开 help window 窗口。

```
②doc doc
```

打开 help window 窗口并定位到 doc 命令的帮助信息。

```
③doc eig
```

则会在工作区中显示：

```
Overloaded methods
doc control/eig
doc symbolic/eig
```

说明 eig 函数已被重载，在 control 及 symbolic 目录下都存在，要想具体定位其中一个函数，可以输入其下面给出的两个命令之一，如：doc control\eig，就可以定位到 Control 目录下的 eig 函数。

3. 显示 M 文件、MAT 文件和 MEX 文件的目录列表

在 MatLab 中，可以用 what 命令来获得 M 文件、MAT 文件和 MEX 文件的目录列表。其基本用法为：what dirname 或 what('dirname')。

如果只输入 what，MatLab 默认显示当前目录下的 M 文件、MAT 文件和 MEX 文件。

在 What dirname 中可显示 dirname 路径下的所有的 M 文件、MAT 文件和 MEX 文件。在此同样不必输入路径的全名，只要输入能使 MatLab 定位的足够的路径段即可。

W=what('bin')　返回给变量 W 一个结构列阵，此列阵的各字段的说明如表 1.3 所示。

表 1.3　结构列阵字段类型及说明

变　量	类　型	说　明
Path	path to directory	dirname 路径的全名
M	cell array of M-file names	M 文件列阵
MAT	cell array of MAT-file names	MAT 文件列阵
MEX	cell array of MEX-file names	MEX 文件列阵
MDL	cell array of MDL-file names	MDL 文件列阵
P	cell array of P-file names	P 文件列阵
Classes	cell array of class names	类列阵

例 7

```
①what
M-files in the current directory F:\MATH SOFTWARES\MatLab\bin
flow_pic    heat4_44    heat4_48    mec2        qqq
yl          ……
MAT-files in the current directory F:\MATH SOFTWARES\MatLab\bin
flow_pic  MatLab
MEX-files in the current directory F:\MATH SOFTWARES\MatLab\bin
OLEAUT32  fmx      libmatlb  libmx    mipcole  mt7s110  uiw
clbs110   glren    libmcc    liboem   mkernel  mwoles05
w32ssi  ……
②what general
M-files in directory F:\MATH SOFTWARES\MatLab\toolbox\MatLab\general
dbstep         isieee         nnload         whatsnew
dbstop         isppc          notebook       which
 ⋮               ⋮              ⋮              ⋮
MEX-files in directory F:\MATH SOFTWARES\MatLab\toolbox\MatLab\general
find_netscape ibrowse      matver       simver
getprofl      iebrowse     rtwver       wrtprofl
P-Files in directory F:\MATH SOFTWARES\MatLab\toolbox\MatLab\general
helpwin
Classes in directory F:\MATH SOFTWARES\MatLab\toolbox\MatLab\general
char
③w=what('general')
w =
       path: 'F:\MATH SOFTWARES\MatLab\toolbox\MatLab\general'
          m: {89x1 cell}
        mat: {}
        mex: { 8x1 cell}
```

```
        mdl: {}
          p: { 1x1 cell}
    classes: { 1x1 cell}
```

“{ }”里列出了各字段的矩阵的大小，可以像引用一般的字符串矩阵一样引用各个字段的矩阵，如：

```
length(w.m)            a=w.p
 ans =                 a =
   89                  'helpwin.p'
```

4. 打印输出一个文本文件的内容

如果要在 MatLab 的工作空间中打印输出一个文本文件的内容，可以用 type filename 命令。此命令等同于 dos 命令中的 type 命令，需给出文件的绝对路径或相对路径，路径的格式等同于操作系统的路径的一般格式。如果没有指定文件扩展名，MatLab 会自动加上.m 的扩展名。如果没有指定路径，MatLab 会按照其默认的搜索路径寻找文件。

例 8　type myfile.txt

打印 myfile.txt 文件内容。

Type humps

MatLab 会打印 humps.m 文件。

5. 在所有 help 条目中搜索指定的关键字

如果要在所有的 help 条目中搜索含有指定关键字的函数和文件，可以用 lookfor 命令，lookfor topic 会在所有 MatLab 搜索路径中的所有 M 文件的第一注释段(关于什么是第一注释段，1.4.1 节中已有介绍)中搜索指定的关键字 topic，并且返回所有在第一注释段中含有此关键字的文件名及包含此关键字的注释行。lookfor topic –all 则指定 MatLab 在所有注释行查找关键字，而不只是在第一注释段中查找，这种查找速度较慢。

例 9　lookfor humps

```
HUMPS  A function used by QUADDEMO, ZERODEMO and FPLOTDEMO.
lookfor humps -all
FUNFUNS Demonstrates functions that operate on other functions.
HUMPS  A function used by QUADDEMO, ZERODEMO and FPLOTDEMO.
FPLOT  Plot function.
```

一般加-all 参数能搜索到更多的条目。

6. 定位函数和文件

函数 which 可以用来定位指定的函数和文件，这在得知函数或文件的文件名但不知道其具体位置时非常重要。其基本的用法有以下 6 种：

```
①which fun
②which fun -all
```

```
③which file.ext
④which fun1 in fun2
⑤which fun(a,b,c,...)
⑥s = which(...)
```

语句①显示指定的 fun 的全部路径及文件名。fun 可以是 M 文件、MEX 文件、工作区的变量、内置函数或 SIMULINK MODEL，对于后 3 种情况，MatLab 会分别显示信息指出 fun 为变量、MatLab 内置函数及 SIMULINK 的一部分。可以在 fun 参数中加入路径以加快定位速度。

语句②显示所有名为 fun 的函数名及路径。返回的第 1 条通常为有 which 命令返回的那一条，其他的或是 shadowed，或是只有在特定的环境中才可以被执行。参数-all 可以用在 which 命令的所有形式中。

语句③显示有指定的文件名及扩展名的文件的整个路径及名称。

语句④显示在 M 文件 fun2 中引用的函数 fun1 的整个路径，等同于调试 fun2 时的 which fun1 命令。可以用此命令验证在一个函数中调用名为 fun1 的函数时，到底是哪个路径下的 fun1 函数被调用。

语句⑤显示有指定参数的函数的路径。

语句⑥将查询的返回结果存入变量 s 中而不是输出到屏幕上，可以通过调用变量 s 来随时查看返回结果。

例 10 which eig

```
eig is a built-in function.(eig是一个内置函数)
which eig -all
eig is a built-in function.
F:\MATH SOFTWARES\MatLab\toolbox\control\@lti\eig.m      % lti method
F:\MATH SOFTWARES\MatLab\toolbox\symbolic\@sym\eig.m     % sym method
F:\MATH SOFTWARES\MatLab\toolbox\MatLab\matfun\eig.m     % Shadowed
```

可以发现，which 不单显示 eig 是内置函数，而且显示了 MatLab 搜索路径中的所有名为 eig 的函数。

在 bin 目录下建立名为 sample.txt 的文本文件，然后输入以下内容：

```
which sample
sample not found.
which sample.txt
F:\MATH SOFTWARES\MatLab\bin\sample.txt
```

不难发现，MatLab 默认的扩展名为.m，要想查找其余扩展名的文件，就必须输入其扩展名。

```
which solve in heat2
F:\MATH SOFTWARES\MatLab\toolbox\symbolic\solve.m
```

则说明，heat2.m 中调用的 solve 函数的路径是 F:\MATH SOFTWARES\MatLab\toolbox\symbolic\solve.m。

```
which feval(inline('sin(x)'))
```

```
    F:\MATH  SOFTWARES\MatLab\toolbox\MatLab\funfun\@inline\feval.m  %
inline method
    which feval('sin(x)')
    feval is a built-in function.
```

上述结果说明，当函数 feval 的参数为 line('sin(x)')时，feval(line('sin(x)'))执行时调用的是函数 F:\MATH SOFTWARES\MatLab\toolbox\MatLab\funfun\@inline\feval.m % inline method；而当其参数为'sin(x)'时，即执行语句 feval('sin(x)')时，调用的是 MatLab 的内置函数。

```
s=which('humps')
s =
F:\MATH SOFTWARES\MatLab\toolbox\MatLab\demos\humps.m
```

返回结果存入 s 变量中，可以调用 s 变量对结果进行处理。

1.4.1 节讨论了几个查找函数或文件的命令：what，which，lookfor。总体说来，what 列出给定目录下的函数；which 用来给出包含给定函数或文件的路径；而 lookfor 用来给出于某个关键字相关的所有路径下的所有函数。它们各有特长，要会灵活运用。

7. 显示演示结果

MatLab 版本中有一个演示程序，分为几大功能，对每个演示程序都会给出所用到的代码以及演示结果。其主要作用是使用户快速了解 MatLab 的各种功能和工作过程。Demo 有 2 种格式：

```
demo ('MatLab'|'toolbox'|'simulink'|'blockset'|'stateflow')
demo argument
```

直接输入 demo 则打开并运行 MatLab 的演示程序。第 1 种情况，参数为“( )”中的任一个即可打开并显示相应的子主题，第 3 种情况用 argument 指定特定的工具箱或类目。

例 11 demo('MatLab')

```
demo MatLab language
```

打开演示程序并将 MatLab 子类展开。

打开演示程序，将 MatLab 子类展开，并将 language 部分反显。读者可以自己试验一下该命令的用法。

8. MatLab 的搜索路径操作

关于 MatLab 的搜索路径的各种操作，在界面操作中已经有所介绍，在命令窗口中主要通过 path(路径操作)、addpath(增加路径)、rmpath(删除路径)3 个命令来实现对路径的各种操作。下面分别介绍这 3 个命令的用法。

(1) path 命令的用法有：

```
p = path
path('newpath')
```

```
path(path,'newpath')
path('newpath',path)
```

path 将 MatLab 中的所有搜索路径输出到屏幕上。关于路径的信息被保存在 toolbox/local/pathdef.m 文件中，虽然这是个文本文件，而且系统允许手工更改此文件，但为了安全起见，建议不要随意更改此文件，最好使用其自带的命令来更改路径。P=path 将所有返回结果存入变量 p 中。Path('newpath') 将当前路径更改为 newpath；path(path,'newpath')将 newpath 路径追加到当前搜索路径的后面；path('newpath',path) 将路径 newpath 加到当前搜索路径的最前面。

MatLab 有一个默认的搜索路径，当输入一个变量或函数名时，如，输入 vname，MatLab 的搜索过程如下：先将其作为一个变量来搜索，如果没有找到，再将其作为一个内置函数来搜索；如找到了，就停止搜索，如还没找到，则在当前路径中寻找 vname.m 和 vname.mex 文件。

如没找到其中的任何一个文件，则在 MatLab 的所有路径中搜索此两个文件，直到找到这 2 个文件之一为止，如仍未找到，则返回找不到文件或变量的信息。

例 12　path

```
MatLabPATH
F:\MATH SOFTWARES\MatLab\bin\myfun
F:\MATH SOFTWARES\MatLab\bin
⋮        ⋮        ⋮        ⋮
```

显示当前的所有搜索路径。

```
path(path,'f:\xb')
path
```

显示：

```
⋮        ⋮        ⋮        ⋮
    F:\MATH SOFTWARES\MatLab\toolbox\database\dbdemos
    F:\MATH SOFTWARES\MatLab\toolbox\java\MatLab
    f:\xb
```

可见已将 f:\xb 追加到搜索路径的后面。

```
path('f:\xb',path)
path
    MatLabPATH
    f:\xb
    F:\MATH SOFTWARES\MatLab\bin\myfun
    F:\MATH SOFTWARES\MatLab\bin
    ⋮        ⋮        ⋮        ⋮
```

可见 f:\xb 已经加到了最前面。

(2) addpath 命令的用法有：

```
addpath('directory')
addpath('dir1','dir2','dir3',···)
```

```
addpath('…','-flag')
```

addpath('dirrectory') 将路径 directory 加到搜索路径的最前面；(addpath 'dir1', 'dir2', 'dir3', …) 将所有指定的路径加到搜索路径的最前面。可以用 path 命令查看运行结果。Addpath…, -flag 将目录列表加到搜索路径的前面或后面，(决定于 flag 的值，如 flag 为 begin，加到前面；如果 flag 为 end，则加到后面)。

例 13　addpath c:\windows –begin

```
addpath c:\windows -end
addpath c:\windows c:\temp -begin
```

(3) rmpath 命令的用法有：

```
rmpath directory/'directory'
rmpath('directory')
```

例 14

```
rmpath c:\windows
rmpath 'c:\temp'
rmpath('c:\windows')
```

1.4.2　管理变量和工作空间

此部分命令主要是用来获得当前工作空间中的变量的信息和对变量进行管理。比较常用的有 who，load，save，size，length 等。

1. 列出当前工作空间中的变量

如果在运算过程中，生成的变量的数目太多，就需要经常获得当前工作空间中的变量的信息，who 和 whos 命令可以用来得到各个变量的简短或详细信息。这 2 个命令的用法列出如下：

```
who
whos
who global
whos global
who -file filename
whos -file filename
who ... var1 var2
whos ... var1 var2
s = who(...)
s = whos(...)
```

who 默认将内存中的当前变量以简单方式列出。whos 默认列出当前内存的变量的名称、大小、是否为复数等信息；who global 和 whos global 列出全局工作空间的变量；who –file filename 和 whos –file filename 列出 MAT 文件中的变量。who var1 var2…和 whos var1 var2…只显示指定的变量 var1 var2…。通配符“*”可以配合用来显示符合某种特定条件的当前工作区中的变量，如 who a*显示当前工作区中以 a 开头的所有变量的信息。Whos('-file',

'filename', 'v1', 'v2'…) 显示 filename 的工作区中的变量 v1, v2, …s=who(…), s=whos(…)将结果赋值给变量 s。

例 15

```
whos('-file','flow_pic.mat','mat8')
  Name      Size         Bytes  Class
  mat8      1x101          808  double array
Grand total is 101 elements using 808 bytes
who
Your variables are:
a         b         c         s         x
s=whos('a')
s =
     name: 'a'
     size: [1 10]
    bytes: 80
    class: 'double'
who -file flow_pic
Your variables are:
mat0      mat20     mat32     mat44     mat56     mat68
 ⋮           ⋮           ⋮         ⋮        ⋮          ⋮
```

2. 变量的文件操作

在对大量的矩阵进行运算操作时，时常需要保存变量或读出变量，这就要用到变量的文件操作。load 可以从文件中恢复以前保存过的变量；save 可以保存工作空间中的变量到指定的文件。

(1) load 没有参数时，系统默认从 MatLab.mat 工作空间文件装入内存。

- load fanme 将名为 fname 的工作空间文件装入内存，如要装入的文件在搜索路径中，可以不输入路径，否则要输入路径。
- load fname x y z 只将 fname 文件中的 x y z 变量装入内存，这里可以用通配符*来匹配符合一定条件的变量。
- load fname.ext 读取一个文本文件，文件中含有与矩阵格式类似的用空格隔开的数据，文件里可以含有以%开头的注释行。其数据将保存在与文件同名的矩阵变量中。
- load fname –ascii/–mat 强制 MatLab 把 fname 当作文本文件或当作工作空间文件。
- s=load(…) 将工作空间文件中的变量返回给结构体变量 s 而不是装入到内存中，s 的各段的名称即是文件中各变量的名称。如果文件是文本格式，s 便是双精度的矩阵。

例 16 a=[]

```
a =
     []
load MatLab.mat a
```

```
a
a =
     1     2     3     4     5     6     7     8     9    10
s=load

Loading from: MatLab.mat
s =
    a: [1 2 3 4 5 6 7 8 9 10]
    b: [ 1x10 double]
    c: 3 + 4i
    x: [10x10 double]
    s: [ 1x1  struct]
s.a
ans =
     1     2     3     4     5     6     7     8     9    10
```

建立文本文件sapmle2.m，内容为：

```
1 2 3 4
5 6 7 8
d=load('sample2.m')
d =
     1     2     3     4
     5     6     7     8
```

(2) save　保存当前工作空间中的变量到 MatLab.mat。

- save fname　将当前工作区中的所有变量保存到 fname 文件中。可以通过 load 命令来恢复数据。
- Save fname x y z　只将变量 x y z 保存到二进制的工作区文件(MAT 文件)中，此处也可以用通配"*"来匹配一类变量。

SAVE fname X Y Z　-ASCII 用 8 位 ASCII 格式保存。

SAVE fname X Y Z　-ASCII -DOUBLE　用 16 位 ASCII 格式保存。

SAVE fname X Y Z　-ASCII -DOUBLE -TABS　用制表符定界。

SAVE fname X Y Z　-V4　保存一个 MatLab 4 版本可以调用的 MAT 文件。

SAVE fname X Y Z　-APPEND　向已存在的 MAT 文件中加入变量。

3. 从内存中清除变量和函数

如果 MatLab 的工作空间中存有许多已经不用的变量或函数，就会占用很多的内存资源，严重时，有可能导致死机。因而经常用函数 clear 将用不到的变量和函数从内存中清除是很有必要的。clear 命令的格式及用法如下：

clear

clear name

clear name1 name2 name3　将 name1，name2，name3 从当前工作空间中清除。

clear global name　将全局变量 name 清除。

clear keyword　这里，keyword 是 functions，variables，mex，global，all 其中之一，其作用分别是将当前工作空间中的正在编译的 M 文件、正在使用的变量、MEX 文件、全局变量、工作区中的所有的变量函数及 MEX 文件清除，all 等于将当前工作区完全清空。

clear 默认将工作区中的所有变量清除。clear name 只是将名为 name 的变量、M 文件、MEX 文件清除。如果变量 name 是全局变量，它只是从当前工作区中清除掉，但在其他的将它声明为全局变量的函数中仍可以访问。如果变量已用 clock 锁住，那么它仍将留在内存中。在选择性清除时，可以使用通配符“*”来匹配符合条件的条目。以上语句也可以采用函数的形式，如 clear('name')。

例 17　clear global a

```
who
Your variables are:
b         c         x         y
clear b
who
Your variables are:
c         x         y
clear global
who
Your variables are:
x         y
clear
who
```

无返回结果。

4. 整理工作空间的内存

pack [filename]　通过压缩信息到最小的内存需求来释放内存。如果不带参数，默认会将变量等信息存入 pack.tmp 中，否则存入 filename 文件中。

这条命令对硬件配置不太好、经常因内存不足而死机的机器非常有用。这条命令并不影响分配给 MatLab 的总内存数量。MatLab 以堆方式管理内存，由于长时间对 MatLab 进行操作，经常导致其内存产生许多碎片，虽然从总体数量上看来，内存还很多，但并没有足够连续的内存空间，从而使得需要较大内存的变量不能得到存储。pack 命令可先将工作空间的所有变量等信息存到硬盘上，然后将内存清空，再从临时文件中取出变量的值，并将临时文件删除，从而达到整理内存的目的。此外，在 Windows 操作系统下，可以通过增大虚拟内存的方法来减少整理内存的次数。

因为此命令要向硬盘中写入临时文件，因而在用此命令时应保证当前的路径是否允许写操作，否则会出错。可以通过编辑一个 M 文件的方法使这个操作过程简化，如同执行批处理命令。先在 MatLab 的搜索路径下建立一个名为 pack2.m 的 M 文件，内容如下：

```
cwd = pwd;
cd(tempdir);
pack
```

```
cd(cwd)
```

然后执行下列语句：

```
echo on
pack2
cwd = pwd;
cd(tempdir);
pack
cd(cwd)
```

此文件的执行过程如下：先打开显示方式(echo on)，将当前路径保存在变量 cwd 中(cwd=pwd；)，将当前工作路径改到 windows 的临时文件路径(cd(tempdir))，然后执行 pack 命令(pack)，再将路径改回到刚才的路径(cd(cwd))。

5. **获得矩阵的尺寸**

矩阵的尺寸，即矩阵的每一维数的长度可以用函数 size()来求得，此函数用法有很多形式：

```
d = size(X)
[m,n] = size(X)
m = size(X,dim)
[d1,d2,d3,···,dn] = size(X)
```

d=size(X)　将多维矩阵 X 的每一维的大小构成的行向量返回给变量 d，d 的元素个数即是矩阵 X 的维数，可以用函数 ndims(X)来求得。[m，n]=size(X)　表示将矩阵 X 的各维的大小返回给 m，n；m=size(X，dim)　表示将矩阵 X 的第 dim 维的大小返回给变量 m；[d1，d2，d3，···，dn]=size(X)　将矩阵每个维的大小返回给左边列出的各单独的变量列表。当左边的变量个数不等于矩阵的维数时，设矩阵的维数为 xm，要赋值的变量共有 n 个，如 n>xm，则矩阵的 xm 个维数的大小将赋值给前 xm 个变量，其余变量的值为 1；如果 n<xm ，前 n-1 个变量将按常规赋值，而最后一个变量将被赋值为所有未赋值的剩余各维的大小的乘积。

```
x=rand(2,3,5)
a=size(x)
a =
     2     3     5
[m1,m2,m3]=size(x)
m1 =                        m2 =                m3 =
     2                           3                   5
[m1,m2]=size(x)
m1 =                        m2 =
     2                          15
[m1,m2,m3,m4,m5]=size(x)
m1 =                        m2 =                m3 =
     2                           3                   5
m4 =                        m5 =
     1                           1
```

6. 获得向量的长度

N=length(x) 此命令完全等价于 n=max(size(x))，即如 x 为向量，返回向量的长度，如 x 为矩阵，则返回 x 的最大的维数。

例 18 x=1:10

```
x =
     1     2     3     4     5     6     7     8     9    10
length(x)
ans =  10
y=rand(2,4,7)
length(y)
ans =    7
max(size(y))
ans =    7
```

7. 在不显示变量名称的同时显示变量的内容

disp(x) 其效果等同于在工作区中直接输入变量 x，而显示的结果只是没有“x=”的字样，在 M 文件中可以用来格式化显示变量的值。

例 19 disp('   col1   col2   col3')

```
disp(rand(2,3))
       col1          col2           col3
       0.13889       0.19872        0.27219
       0.20277       0.60379        0.19881
```

1.4.3 文件及操作系统命令介绍

此部分的命令主要用来对 MatLab 下的目录和文件进行管理和操作，其命令执行的格式和各命令的意义与传统的 DOS 命令都有着许多的相似之处，如改变当前路径的 cd 命令、显示当前目录下的子目录和文件的列表的 dir 命令等都几乎是完全相同的，而且也可以使用通配符。

1. 改变当前路径

如果要改变 MatLab 的当前的工作路径，可以用 cd 命令，此命令等同于 DOS 操作系统的命令 cd。如果不带参数，默认会显示当前的路径，cd directory 将路径改变到 directory，如果想进入当前目录的子目录，可以只输入目录名，否则要输入全路径。cd.. 改变当前路径到上一级目录。

例 20 cd

```
F:\MATH SOFTWARES\MatLab\bin
cd ..
cd
F:\MATH SOFTWARES\MatLab
cd bin
```

```
cd
F:\MATH SOFTWARES\MatLab\bin
```

2. 显示当前目录下的子目录和文件列表

在 MatLab 中可以直接使用操作系统的命令 dir 来显示当前目录下的子目录和文件列表。dir dirname 显示特定目录下的文件，可以使用所在操作系统中的全部与 dir 语句有关的通配符*的操作。name=dir('dirname')或 name=dir 将返回结果以 m×1 的结构体的形式赋给变量 name，各字段的名称为：name，文件名称；date，修改时间；bytes，文件的大小；isdir，如果 name 对应的是目录，则为 1，否则为 0。

```
cd
F:\MATH SOFTWARES\MatLab
dir
.           DeIsL2.isu   extern       java        simulink
..          bin          ghostscript  relnotes.txt stateflow
DeIsL1.isu  exlink       help         rtw         toolbox
s=dir('bin')
s =
107x1 struct array with fields:
    name
    date
    bytes
    isdir
```

3. 删除文件和图形句柄

在 MatLab 的命令窗口中也可以执行删除文件的操作，删除文件的命令为 delete。delete filename 删除名为 filename 的文件，在这里可以使用通配符*，如 delete *.m 删除当前目录下所有的 M 文件。如果文件名存放在字符串变量 s 中，可以使用 delete 的函数形式 delete(s)。这个函数不会显示出确认信息，所以要谨慎使用，以防误删文件。

delete 命令也可以用来删除图形对象。delete(h) 删除句柄为 h 的图形对象。这一操作会将图形窗口删除并关闭，而不会显示任何确认信息。

4. 获取环境变量的值

MatLab 中有几个环境变量，环境变量的值随着不同的机器有所不同。如果需要得到环境变量的值，可以用 getenv 函数，其调用格式为 getenv('name')，其中，name 是一个字符串，为要获取的环境变量的名称。此函数以 name=value 的形式返回一系列列表，如果对应的环境变量 name 未找到，则将返回空值。

例 21 getenv('path')

```
ans =
C:\WINDOWS;C:\WINDOWS\COMMAND;F:\MATH;SOFTWARES\MatLab\BIN;E:\PFW;C:
\WINDOWS\TWAIN_32\SCANPORT;C:\WINDOWS\TWAIN\SCANPORT
```

5. 执行操作系统的命令

操作符!可以在 MatLab 的命令窗口下执行操作系统的命令，如各种 DOS 命令、可执行程序等，命令执行完毕后，会自动返回到 MatLab 的命令窗口。

例 22　!dir c:*.exe

```
Volume in drive C is SYSTEM
Volume Serial Number is 1665-1BE0
Directory of C:\
INST     EXE       245,108  05-14-99  16:42 INST.EXE
         1 file(s)       245,108 bytes
         0 dir(s)     205,631,488 bytes free
```

!c:\inst 可以执行这个 EXE 文件。

此外，UNIX 命令可以执行操作系统命令并返回执行结果。

[status，result] = UNIX('command')，一般 status 返回 0，而 result 保存返回程序执行的结果，请参照下面的例子。

```
[s,w]=unix('dir D:\')
s =
    0
w =
Volume in drive D is 大型软件箱
Volume Serial Number is 6D5B-B4B2
Directory of D:\
AUTOCAD          <DIR>        02-27-99  23:06 AutoCad
VISUAL~1 0       <DIR>        01-28-99  18:41 VISUAL FOXPRO 5.0
PROGRA~1         <DIR>        03-07-99  17:33 Program Files
金山词~2          <DIR>        02-27-99  22:32 金山词霸-R3
MICROS~2         <DIR>        01-28-99  20:50 Microsoft Office 97
PHOTOS~1         <DIR>        03-08-99  19:03 photoshop
PAINTS~1         <DIR>        03-12-99  16:24 Paint Shop Pro 5
OFFICE           <DIR>        05-09-99  21:35 OFFICE
MATHSO~1         <DIR>        04-02-99  21:34 MATH SOFTWARES
         0 file(s)             0 bytes
         9 dir(s)     361,500,672 bytes free
```

6. 建立日志文件来保存 MatLab 的任务

关于日志文件也许并不陌生，如在 UNIX 系统中，日志文件主要用来记录登陆系统的操作过程以及系统的一些运行状态，便于管理员分析和管理系统，维护系统正常安全的运行。在 MatLab 中的日志文件功能没有这么强大，但还是有一定相似之处的。命令 diary 可以用来建立日志文件，其调用格式如下：

```
diary
diary filename
diary off/on
```

此命令以文本文件的方式记录在工作区内所作的键盘输入以及系统的回应。如不指定文件，则在当前目录下保存在默认文件 diary 里。可以用 diary filename 指定要保存到的目的文件名，文件名可以为除“on”和“off”以外的所有合法的文件名，将除图形以外的显示信息保存下来，以便参考和处理。如果指定的文件已经存在，则会继续向此文件追加数据而不是将此文件替换。可以用 diary 及 diary on/off 命令来激活或挂起日志文件的记录。此命令的函数形式是 dirary('filename')。

例 23 diary

```
diary xb
dir
.       diary   gradF.m pack2.m xb.mat
..      grad.m  gradV.m xb
```

在文件列表中，没有后缀的 diary 文件和 xb 文件即是刚才所建立的两个日志文件。

1.4.4 控制窗口的命令

此部分的命令主要用来对命令窗口的显示方式进行操作，如设置命令行编辑、设置输出分页等。下面给出这些设置的简单说明。

1. 设置命令行编辑

CEDIT('off') 关闭命令行的回显。
CEDIT('on') 打开命令行的回显。
CEDIT('vms') 选择按键定义。
CEDIT('emacs') 恢复系统默认的键定义。
下面将给出在命令行中用到的有用按键的定义。
前一条命令 ^p (preview)
下一条命令 ^n (next)
向左移动一个字符的位置 ^b(back)
向右移动一个字符的位置 ^f (forward)
删除整行命令 esc
将光标移到所在字段的左边 ^l (left)
将光标移到所在字段的右边 ^r (right)
将光标移到行的开始 ^a
将光标移到行的结尾 ^e
取消一行的输入 ^u (undo)
删除当前字符 ^d (delete)
删除从光标位置到行末的所有字符 ^k
以上命令很简单，只要亲手使用一下即可学会，在此不再举例说明。

2. 清除命令窗

clc 将命令窗口中的所有显示字符全部删除，同时光标移至左上角。

3. Home 将光标返回到窗口的左上角

例 24 clc

```
home
```

4. 控制窗口命令行输出分页

MatLab 中的输出分页形式的设置是由 more 命令来实现的，如同在 DOS 状态下使用 dir 命令时加了/p 参数。

more off　　关闭命令行窗口的分页输出形式。

more on　　打开命令行窗口的分页输出形式。

more(n)　　设置分页输出形式为每页输出 n 行。

当设置了分页输出形式，且 MatLab 需要分页输出时，会显示--more--等待命令，以显示下一条信息或下一页记录。此时，如果按 Enter 键就会显示下一条要输出的信息；如果按 Space 键则会显示下面一整页的输出信息；按 Q 键会中断此次文本显示。

例 25

```
more on                        %打开分页功能
more off                       %关闭分页功能
more(10)                       %将每页显示限制在 10 行
path
        MatLabPATH
    F:\MATH SOFTWARES\MatLab\bin\myfun
    F:\MATH SOFTWARES\MatLab\bin
    F:\MATH SOFTWARES\MatLab\toolbox\MatLab\datafun
    F:\MATH SOFTWARES\MatLab\toolbox\MatLab\datatypes
    F:\MATH SOFTWARES\MatLab\toolbox\MatLab\demos
    F:\MATH SOFTWARES\MatLab\toolbox\MatLab\elfun
    F:\MATH SOFTWARES\MatLab\toolbox\MatLab\elmat
--more--                       %等待按键
```

1.4.5 一般信息获得

info 命令获得关于 MatLab 的信息和 MathWorks 公司的信息，此信息包括 MatLab 软件可以应用的机型、MathWorks 公司的产品介绍、及其公司的联系地址和方法等。

Ver　得到 MatLab，simulink 和其 toolbox 的版本信息。

Hostid　得到 MatLab 服务对象的识别号。

Whatsnew　得到 MatLab 最近比较新的信息。

Whatsnew toolboxpath　显示指定的工具箱的 Readme 文件。

例 26

```
whatsnew MatLab               %显示 MatLab 的 Readme 文件
whatsnew signal               %显示 MatLab 信号处理工具箱的 Readme 文件
```

1.4.6 启动和退出 MatLab

在启动 MatLab 时，会自动执行主启动 M 文件 MatLabrc.m，如果文件 startup.m 存在，则进而执行 startup.m 文件，然后才正式进入操作阶段。因此，如果想在 MatLab 启动时让它自动执行某条命令或某个文件，或自动加载某些自定义函数，或自动定义一些特定的有用的常量、变量等，可以编辑 startup.m 文件，在此文件内完成想让 MatLab 自动完成的命令，这样可以避免每次都进行同样的重复性劳动，节省了时间和精力。

注意：

- startup.m 文件必须在 MatLab 的搜索路径里，否则 MatLab 将找不到文件。
- 如果对多用户系统或网络用户，MatLabrc.m 文件只对系统管理员开放。
- Quit 终止并退出 MatLab 的运行环境，退出时并不保存工作空间，因而建议在使用此命令之前应确认是否需要保存工作空间，如需要，则应用 save 命令进行保存，请参照“管理变量和工作空间”部分。

1.5 帮助的使用及其在线信息的利用

MatLab 是功能强大的应用软件，具有许多内置函数和功能完备的工具箱，各种命令和函数数以百计，而每条命令都有自己独特的参数，每个函数也都有不同的参数和返回结果。面对这么多的函数和功能，每个人都不可能精通每个函数和命令，即使对那些常用的命令和函数亦是如此。那么当需要某个函数而又不了解它的具体用法时，一个办法就是查手头的资料，这通常是不太现实的，即使有合适的资料供查找，也很费时间；另一个办法就是利用 MatLab 的在线查找功能。MatLab 提供了非常完备的在线查找功能，在使用过程中，可以发现，理解、掌握和精通这种方法是非常必要和有效的。对于查询系统的调用方式而言，有 3 种方式：

- 从 MatLab 的帮助窗口中获得帮助信息。
- 在 MatLab 工作空间的命令行中直接键入帮助命令。这种方法直接了当，获得信息简洁而迅速，最为常用。
- 利用 MatLab 提供的强大的在线帮助桌面。这里面提供了按字母排序的指令索引表和按内容排序的指令索引表，还提供了功能强大的搜索引擎，在线查找各类信息，并可以运用逻辑关系运算，功能最为强大。

这 3 种方式各有特色，灵活运用一般都能查到所需要的信息，下面将分别介绍这 3 种查找方式。

1.5.1 从 MatLab 的帮助窗口中获得帮助信息

单击 Help|Help Window 或 Help|Help tips 命令即可启动 Help Window 窗口。界面如图 1.17 所示。

在左上角的文本框内可以输入要查找的主题，然后按 Enter 键即会在下面的文本框内显示相关主题的查找结果。右边的 See also 下拉列表框显示与当前查找的主题相关的其他

主题，可以双击某个条目来显示这个主题的内容，以便于充分了解某个命令和函数。单击Back按钮可以回到上一次显示的内容，单击Forward按钮可以跳到下一个显示内容，单击Home则会回到图所示的显示结果。单击Go to Help Desk按钮则会打开MatLab帮助桌面，如右图中，在显示文本框内双击某个条目亦可以显示相关的帮助主题的信息。下面举例来说明。

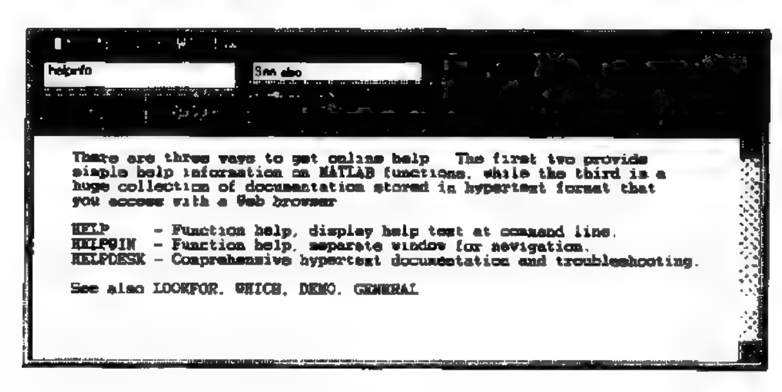


图1.17 MATLAB Help Window窗口

例27 在主题文本框内输入fourier，则会显示如下：

```
--- help for sym/fourier.m ---
 FOURIER Fourier integral transform.
    F = FOURIER(f) is the Fourier transform of the sym scalar f
    with default independent variable x.  The default return is
    a function of w.
⋮                       ⋮                       ⋮                       ⋮
```

双击See also下拉列表框，然后双击其中的ifourier选项，则在显示文本框内会马上显示如下：

```
--- help for sym/ifourier.m ---
 IFOURIER Inverse Fourier integral transform.
  f = IFOURIER(F) is the inverse Fourier transform of the scalar sym F
    with default independent variable w.  The default return is a
⋮                       ⋮                       ⋮                       ⋮
```

单击Back按钮则会重新显示fourier的帮助信息，再单击Forward按钮则又会显示ifourier的帮助信息。

1.5.2 在MatLab工作空间的命令行中直接键入帮助命令

在命令行中可以配合使用doc，help，lookfor，exit，which，who，whos，what等命令来对函数、文件、变量和目录等信息进行查找。由于以上命令的绝大多数已经详细介绍过了，在这里只是简单介绍一下怎样利用这些命令来查找需要的信息。

在命令窗口中输入doc function可以直接启动帮助桌面并显示与function有关的HTML格式的帮助信息。如，doc which，即可得到与which命令有关的信息。

如果想知道 eig 的有关信息，可以这样操作：

```
which eig -all
eig is a built-in function.
F:\MATH SOFTWARES\MatLab\toolbox\control\@lti\eig.m        % lti method
F:\MATH SOFTWARES\MatLab\toolbox\symbolic\@sym\eig.m       % sym method
F:\MATH SOFTWARES\MatLab\toolbox\MatLab\matfun\eig.m       % Shadowed
```

如果对第 3 个条目感兴趣，则继续输入：

```
help 'F:\MATH SOFTWARES\MatLab\toolbox\symbolic\@sym\eig.m'
 EIG   Symbolic eigenvalues and eigenvectors.
   With one output argument, LAMBDA = EIG(A) is a symbolic vector
   containing the eigenvalues of a square symbolic matrix A.
    With two output arguments, [V,D] = EIG(A) returns a matrix V whose
   columns are eigenvectors and a diagonal matrix D containing
   eigenvalues.
   If the resulting V is the same size as A, then A has a full set of
   linearly independent eigenvectors which satisfy A*V = V*D.
    With three output arguments, [V,D,P] also returns P, a vector of
    indices
   whose length is the total number of linearly independent eigenvectors,
   so that A*V = V*D(P,P).
    LAMBDA = EIG(VPA(A)) and [V,D] = EIG(VPA(A)) compute numeric
    eigenvalues
   and eigenvectors using variable precision arithmetic.  If A does not
   have a full set of eigenvectors, the columns of V will not be linearly
   independent.
    Examples:
      [v,lambda] = eig([a,b,c; b,c,a; c,a,b])
       R = sym(gallery('rosser'));
      eig(R)
      [v,lambda] = eig(R)
      eig(vpa(R))
      [v,lambda] = eig(vpa(R))
       A = sym(gallery(5)) does not have a full set of eigenvectors.
      [v,lambda,p] = eig(A) produces only one eigenvector.
    See also POLY, JORDAN, SVD, VPA.
```

另如，想了解所有与 fourier 关键字有关的函数的信息，可以输入：

```
lookfor fourier
FFT Discrete Fourier transform.
FFT2 Two-dimensional discrete Fourier Transform.
FFTN N-dimensional discrete Fourier Transform.
IFFT Inverse discrete Fourier transform.
IFFT2 Two-dimensional inverse discrete Fourier transform.
IFFTN N-dimensional inverse discrete Fourier transform.
XFOURIER Graphics demo of Fourier series expansion.
```

```
DFTMTX Discrete Fourier transform matrix.
PLOTFOU Plot contents of Fourier files
EXPFOU Write data to a Fourier vector or a (maybe existing) file (for
ELIS).
IMPFOU Read complex amplitudes from a Fourier vector or file (used by
ELIS).
MODIFYFV Modify Fourier (maybe also variance) data by given transfer
function.
SIMFOU Generate simulated Fourier amplitudes.
INSTDFFT Inverse non-standard 1-D fast Fourier transform.
NSTDFFT Non-standard 1-D fast Fourier transform.
FOURIER Fourier integral transform.
IFOURIER Inverse Fourier integral transform.
```

然后输入想了解的函数名，如 help FFT，即可得到想要的信息。

又如想了解目录 F:\MATH SOFTWARES\MatLab\toolbox\MatLab\polyfun 下都有哪些有用的函数和命令，可以这样操作：

```
what('F:\MATH SOFTWARES\MatLab\toolbox\MatLab\polyfun')
M-files in directory F:\MATH SOFTWARES\MatLab\toolbox\MatLab\polyfun
Contents  icubic    interp5   polyarea  resi2    table1   unmkpp
abcdchk   inpolygon interp6   polyder   residue  table2   xychk
automesh  interp1   interpft  polyfit   roots    tf2ss    xyzchk
convhull  interp1q  interpn   polyval   spline   tf2zp    xyzvchk
delaunay  interp2   mkpp      polyvalm  splncore tfchk    zp2ss
dsearch   interp3   mpoles    ppval     ss2tf    tsearch  zp2tf
griddata  interp4   poly      rectint   ss2zp    tzero
MEX-files in directory F:\MATH SOFTWARES\MatLab\toolbox\MatLab\polyfun
delaunay dsearch  tsrchmx
```

例 28　了解 who 函数的具体用法

```
help who
 WHO    List current variables.
    WHO lists the variables in the current workspace.
    WHOS lists more information about each variable.
    WHO GLOBAL and WHOS GLOBAL list the variables in the global workspace.
    WHO -FILE FILENAME lists the variables in the specified .MAT file.
     WHO ... VAR1 VAR2 restricts the display to the variables specified.
    The wildcard character '*' can be used to display variables that
    match a pattern.  For instance, WHO A* finds all variables in the
    current workspace that start with A.
     Use the functional form of WHO, such as WHO('-file',FILE,V1,V2),
    when the filename or variable names are stored in strings.
     S = WHO(...) returns a cell array containing the names of the
    variables in the workspace or file. You must use the functional
    form of WHO when there is an output argument.
     See also WHOS.
```

从这些信息可以看出，who 命令的在线帮助信息对 whos 命令语句的说明是非常详细的，完全可以在不了解 who 命令的具体用法的基础上，查看其帮助信息来学习 who 命令的具体用法，同时可以在 MatLab 环境下使用，其学习效果更加明显。

还可以试验一下，在命令行直接输入 help whos 命令与在帮助窗口中输入 whos，其显示内容是一样的。

欲了解当前工作空间中所定义的变量的有关信息，可以用 who 及 whos 命令，如需要可以参照 1.4 节通用命令部分，这里不再赘述。

由以上几个例子可以看出，灵活运用上面的几个命令可以得到所需要的大多数信息。足以看出 MatLab 在线帮助功能的强大。

1.5.3 在线帮助桌面

帮助桌面的信息查询是 MatLab 提供的帮助系统中，功能最强、查找范围最广的一种在线帮助，如图 1.18 所示它可以访问当地硬盘及光驱上的帮助信息，还可以通过互联网访问 INTERNET 上的大量资源，其中包括该公司所提供的远程在线帮助站点，可以访问 Mathworks 公司主页查看最新的技术支持；可以通过发 E-mail 的方式提出问题和对此软件的见解和看法，说明其中存在的 bug，与公司进行交流。这些都需要已经联入互联网。所有的帮助信息都是以超文本格式存放的，因而需要安装浏览器，如网景公司的 Netscape 及微软公司的 Internet Explorer 是当前最为流行的 2 种浏览器，使用其中任何一种皆可。

单击 Help|Help Desktop 命令，或在命令窗口中键入 doc 即可打开帮助桌面。打开后界面如图 1.18 所示。其中的每一条显示蓝色的字样就是一条超级连接，单击 subject 即可进入以主题分类的函数列表，单击 Index 即可进入以字母为排列顺序的函数列表。通过这 2 种方法可以通过相关主题或字母来快速定位此函数的帮助信息。

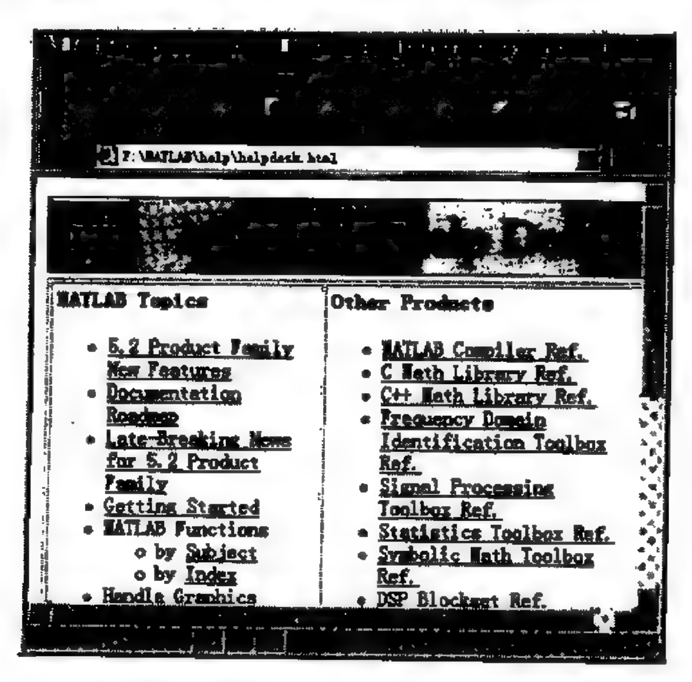


图 1.18 MatLab 在线帮助桌面

例 29

要想查找关于 path 命令的有关信息，可以通过以下步骤达到：

依次单击 by Subject|General Purpose Commands|path 命令即可找到 path 的帮助信息。

依次单击 by Index|P|Path 亦可以找到 path 命令的信息。

另外在 help desktop 的每一页中都有一个如图 1.19 所示的搜索框，在文本框中输入要搜索的函数的名字即可迅速定位该函数的帮助信息。

如例 29 中，在文本框中输入 path，单击 Go 按钮即可找到 path 的帮助信息。

回到 Help Desk 的主界面，单击 Full-text Search 按钮可以进入全文检索界面。



图 1.19 迅速定位某个函数的帮助文件

如图 1.20 所示，在 Search 下拉列表框中，选择 MATLAB Function Reference 选项，将会在函数索引中搜索信息。

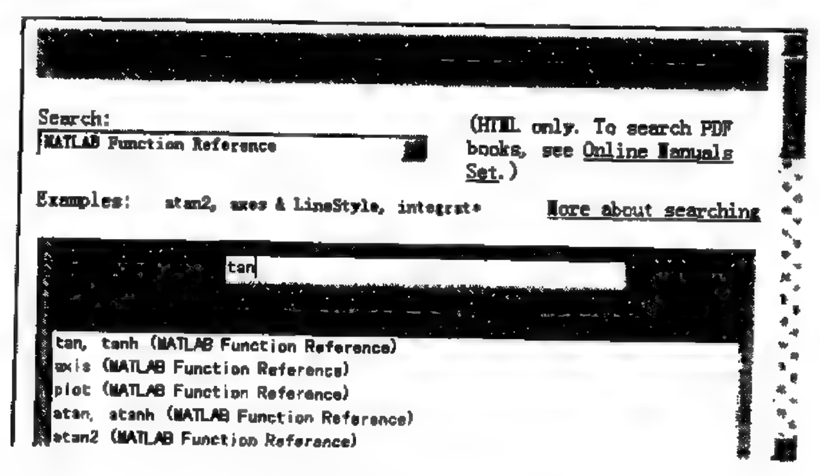


图 1.20 全文检索界面

如果选择 All MATLAB and Other Product Documents 选项，则会在所有的文件中搜索指定的关键字，显然，后一种会搜索到更多的符合条件的条目。

例 30 (1)在 Enter query 文本框中输入 tan，单击 Search 按钮可以得到搜索结果：

```
tan, tanh (MatLab Function Reference)
tan,  tanh(C++ Math Library)
bilinear (Signal Procssing Toolbox)
mlfTan, mlfTanh (C Math Library)
axis (MatLab Function Reference)
Trigonometric Function (DSP Blockset)
Plot (MatLab Function Reference)
Atan2 (MatLab Function Reference)
Atan, atanh( MatLab Function Reference)
```

```
Finvers(Symbolic Math Toolbox)
Trigonometric Function
Fcn
List of Commands (MatLab Function Reference)
```

如果在 Search 下拉列表框中选择 MATLAB Function Reference 选项，则对 tan 关键字只能搜索到如下的信息：

```
tan, tanh (MatLab Function Reference)
axis (MatLab Function Reference)
Plot (MatLab Function Reference)
Atan, atanh( MatLab Function Reference)
Atan2 (MatLab Function Reference)
```

双击以上显示的各条目即可进入相关的帮助信息，如上面双击 plot 项，可以得到关于 plot 函数的帮助信息。

在 Enter Query 字段中输入的查找信息中，可以包含关键字和逻辑操作符。所有的关键字都是对大小写不敏感的，这里支持的逻辑操作符有：

AND(符号为 '+'或'&')　　逻辑与操作

OR (符号为 '—' 或'|')　　逻辑或操作

NOT(符号为'! ')　　逻辑非操作

此外，可以使用通配符*来匹配特定的项。

例 31　查找 poly*可以得到如下结果：

```
polyvalm(MatLab Function Reference)
polyval(MatLab Function Reference)
polyarea(MatLab Function Reference)
polyfit(MatLab Function Reference)
poly(MatLab Function Reference)
polyder(MatLab Function Reference)
residue(MatLab Function Reference)
compan(MatLab Function Reference)
fill(MatLab Function Reference)
fill3(MatLab Function Reference)
roots(MatLab Function Reference)
inpolygon(MatLab Function Reference)
path(MatLab Function Reference)
convhull(MatLab Function Reference)
polyeig(MatLab Function Reference)
voronoi(MatLab Function Reference)
```

查找 poly AND val 得到结果：

```
vicontrol(MatLab Function Reference)
poly(MatLab Function Reference)
roots(MatLab Function Reference)
polyvalm(MatLab Function Reference)
compan(MatLab Function Reference)
```

```
residue(MatLab Function Reference)
```

查找 compan | val AND poly 得到：

```
compan(MatLab Function Reference)
vicontrol(MatLab Function Reference)
poly(MatLab Function Reference)
roots(MatLab Function Reference)
hadamard(MatLab Function Reference)
polyvalm(MatLab Function Reference)
residue(MatLab Function Reference)
```

1.6 习　　题

(1) 以两种不同的方式进入 MatLab 的工作环境，并退出。

(2) 按照第 2 节第 2 部分所介绍的步骤，依次执行各条命令。

(3) 熟悉 MatLab 的各栏菜单的功能和各工具栏的功能。

(4) 在 MatLab 工作空间中建立以下两个矩阵：

```
A=[ 2  4  7          B=[ 1  3  5
    3  1  6]             2  2  5
                         4  8  0]
```

将当前的工作空间保存为 mat1.mat，并关闭 MatLab。

(5) 打开 MatLab，用 who 命令查看当前工作空间中的变量，然后装载 mat1.mat 文件，再用 who 命令查看当前的变量，并查看变量的值。

(6) 查看当前 MatLab 的搜索路径，建立一个新目录，并将其加入 MatLab 的路径中，同时将此目录设为当前工作路径。

(7) 熟练掌握各条通用命令。.

(8) 学会查找和利用在线帮助信息，熟练掌握查找帮助信息的各条命令。

(9) 用 lookfor 查找关于函数 eig 的信息，并打印出 eig 函数的内容。

(10) 在工作空间中建立两个变量 A 和 B，可以为矩阵或字符串，然后用 who 命令查看当前工作空间中的变量及其类型。将变量 A 删除，将变量 B 保存在 DATA.mat 中，然后将所有变量删除，用 load 命令装载 DATA.mat 文件，并查看当前工作空间中的变量。

第 2 章　MatLab 基础知识介绍

MatLab 语言以前是专门为进行矩阵计算所设计的一门语言，在以后的各个版本中逐步扩充其各种功能。现在 MatLab 不仅仅局限于矩阵计算领域，但其最基本、最重要的功能还是进行实数矩阵和复数矩阵的运算。MatLab 语言作为一门主要用于计算的语言，与 C 或 FORTRAN 等高级语言有所不同。在第 2 章中，主要介绍 MatLab 语言及命令的主要基本知识，这是 MatLab 的最基本和最重要的部分，对第 2 章内容的深入理解和掌握是对其他各章节进行理解和运用并对 MatLab 进行扩展的基础。

2.1　一般运算符及操作符

一般运算符和操作符构成运算的最基本的操作指令，如加、减、乘、除和乘方等运算，这些运算指令几乎在所有计算机语言中都有，且大同小异。在 MatLab 中，几乎所有的操作都是以矩阵为基本运算单元的，这与其他计算机语言有很大不同，也是 MatLab 的重要特点，在以后的学习中应该充分理解和注意。

2.1.1　运算符

1. 矩阵的加减运算

其基本形式为 X±Y，X 和 Y 必须为同维数的矩阵，此时各对应元素相加减；或者其中一个为数，即 1×1 矩阵，此时即相当于将 X 或 Y(为 1×1 矩阵的变量)变为各元素都为 X 或 Y、且与另一同维的矩阵再进行加减运算。

```
y =
     2     3
     4     5
x =
     3     4
8
z=x+y
z =
     5     7
13
2-y
ans =
     0    -1
    -2    -3
```

2. 矩阵的乘法运算

X*Y 是两个矩阵 X 和 Y 的乘积，其中 X 和 Y 必须满足矩阵相乘的条件，即矩阵 X 的列数必须等于矩阵 Y 的行数。如果其中一个为 1×1 矩阵亦合法，此时便是将每一个矩阵的元素都分别与这个数值相乘。

例 1　x*y

```
ans =
     22    29
     44    58
3*x
ans =
      9    12
24
```

3. 矩阵的数组乘法

X.*Y，运算结果为两个矩阵的相应元素相乘，得到的结果与 X 和 Y 同维，此时 X 和 Y 也必须有相同的维数，除非其中一个为 1×1 矩阵，此时运算法则与 X*Y 相同。

例 2　x.*y

```
ans =
6    12
40
y.*2
ans =
     4     6
10
```

4. 矩阵的乘方运算

x^Y，X^y

(1) X^y 表示，如果 X 是方阵、y 是一个大于 1 的整数，所得结果由 X 重复相乘 y 次得到；如果 y 不是整数，则将计算各特征值和特征向量的乘方。

```
x^1.5
ans =
       9.9499       13.266
       19.9        26.533
x^2
ans =
     33    44
88
```

(2) x^Y 表示，x 为数，而 Y 为方阵，结果由各特征值和特征向量计算得到。

```
2^x
```

```
ans =
       559.27       744.36
       1116.5       1489.7
```

(3) 如果X和Y都是矩阵，或X或Y不是方阵，则会显示错误信息。

5. 矩阵的数组乘方

X.^Y 的计算结果为 X 中元素对 Y 中对应元素求幂，形成的矩阵与原矩阵维数相等，这里 X 和 Y 必须维数相等，或其中一个为数，此时运算法则等同于 X^Y。

例 3 x=

```
    1    2
    3    4
y =
    2    2
    1    3
x.^y
ans =
    1    4
64
```

6. 矩阵的左除运算

A\B 称作矩阵 A 左除矩阵 B，其计算结果大致与 INV(A)B 相同，但其算法却是不相同的。如果 A 是 N×N 的方阵，而 B 是 N 维列向量，或是由若干 N 维列向量组成的矩阵，则 X=A\B 是方程 AX=B 的解，X 与 B 的大小相同，对于 X 和 B 的每个列向量，都有 AX(n)=B(n)，此解是由高斯消元法得到的。很显然，A\EYE(SIZE(A))=INV(A)EYE(SIZE(A))=INV(A)。如果 A 是 M×N 的矩阵(M≠N)，B 是 M 维列向量或由若干 M 维列向量组成的矩阵，则 X=A\B 是欠定或超定方程 AX=B 的最小二乘解。A 的有效秩 L 由旋转的 QR 分解得到，并至多在每列 L 个非零元素上求解。

例 4 A=[1 2; 2 1]

```
B=[1 2; 1 2]
A\B
ans =
    0.3333    0.6667
    0.3333    0.6667
```

7. 矩阵的右除运算

B/A 称为矩阵 A 右除 B，其计算结果基本与 B*INV(A)相同，但其算法是不同的，可以由左除得到，即：B/A=(A'\B')'。它实际上是方程 XA=B 的解。

例 5 B/A

```
ans =
    1    0
```

```
    1    0
(A'\B')'
ans =
    1    0
    1    0
```

8. 矩阵的点除运算

B./A，如果 B 和 A 都是矩阵，且维数相同，则 B./A 就是 B 中的元素除以 A 中的对应元素，所得结果矩阵的大小与 B 和 A 都相同；如果 B 和 A 中有一个为数，则结果为此数与相应的矩阵中的每个元素作运算，结果矩阵与参加运算的矩阵大小相同。

例 6 B./A

```
ans =
    1.0000    1.0000
    0.5000    2.0000
```

A./2

```
ans =
    0.5000    1.0000
    1.0000    0.5000
```

9. 矩阵的 KRONECKER 张量积

K=KRON(A，B) 返回 A 和 B 的张量积，它是一个大矩阵，取值为矩阵 A 和 B 的元素间所有的可能积。如果 A 是 m×n 矩阵，而 B 是 p×q 矩阵，则 KRON(A，B)是 mp×nq 的矩阵。即，如果 A 是 2×3 的矩阵，则有下式成立：

```
KRON(A, B)=[ A(1,1)*B  A(1,2)*B  A(1,3)*B
             A(2,1)*B  A(2,2)*B  A(2,3)*B ]
```

如果 A 和 B 中有一个为稀疏矩阵，则只有非零元素会参与计算，所得的结果亦是稀疏矩阵。关于稀疏矩阵将在第 4 章讲到。

例 7 kron(A，B)

```
ans =
    1    2    2    4
    1    2    2    4
    2    4    1    2
    2    4    1    2
```

2.1.2 操作符

1. 冒号 “：”

此符号在矩阵的构造和运算中非常有用，它可以用来产生向量，用作矩阵的下标，以及部分地选择矩阵的元素，行循环操作等，熟练掌握可以在矩阵的运算中受益匪浅。其基本用法有：

```
j:k        等价于[j,j+1,.......,k]
j:i:k          等价于[j,j+i,j+2*i,......k]
```

j:k 中，如果 j<k 则返回空值；j:i:k 中，如果 j>k 并且 i>o，或 j<k 并且 i<o 都会返回空

值。

A(:，i) 取 A 矩阵的第 i 列。

A(i，:) 取矩阵 A 的第 i 行。

A(:，:) 以 A 的所有元素构造二维矩阵，如果 A 是二维矩阵，则结果就等于 A。

A(j:k) 等价于 A(j)，A(j+1)，…，A(k)。

A(:，:，k) 三维矩阵 A 的第 3 页。

A(:) 将所有 A 的元素作为一个列向量，如果此操作符在赋值语句的左边，则用右边矩阵的元素来填充矩阵 A，矩阵 A 的结构不变，但要求两边矩阵的元素个数相同，否则会出错。

例 8 a=rand(3，4)

```
a = 0.5678   0.6029   0.3050   0.7680
    0.7942   0.0503   0.8744   0.9708
    0.0592   0.4154   0.0150   0.9901

b=rand(2,2,3)
b(:,:,1) =              b(:,:,2) =               b(:,:,3) =
         0.7889   0.4983          0.6435   0.9601          0.4120
0.2679
         0.4387   0.2140          0.3200   0.7266          0.7446
0.4399
b(:,:,2)
ans =   0.6435   0.9601
        0.3200   0.7266
b(:,:)
ans =   0.7889   0.4983   0.6435   0.9601   0.4120   0.2679
        0.4387   0.2140   0.3200   0.7266   0.7446   0.4399
a(:)=b
a = 0.7889   0.2140   0.9601   0.7446
    0.4387   0.6435   0.7266   0.2679
    0.4983   0.3200   0.4120   0.4399
```

2. 百分号："%"

百分号在 M 文件和命令行中表注释，即在一行中百分号后面的语句都被忽略而不被执行。在 M 文件中，百分号后面的语句可以用 help 命令打印出来。

3. 连续点："..."

如果一条命令很长，一行容不下，可以用 3 个或更多的点加在一行的末尾，表示此行未完，而在下一行继续。

4. 引号："'"

表示矩阵的转置。

5. 分号："；"

分号用在"[ ]"内，表示矩阵中行的结尾；也可以用在每行命令的结尾，则命令不会回显。可以用在 M 文件中控制命令的显示，并压缩输出篇幅。

例 9 c=[1 2;3 4]

输入 c，则显示：

```
c =  1     2
     3     4
```

其余运算符号及特殊符号的功能请参照附录。

2.2 数据格式显示

关于通过 MatLab 的菜单来调整数据格式的显示，已经在第 1 章中介绍过，在这一节中，将着重介绍通过命令行来调整数据的显示格式。

MatLab 的所有计算都是通过双精度进行的，在内存中数的精度都是双精度的，但其显示格式却可以有不同形式。在 MatLab 的命令行中通常用 format 命令在数据的不同显示格式之间切换。format 命令的主要用法有：

(1) format 默认值，数据显示格式与 short 格式相同。

(2) format short 短格式，只显示 5 位数值。对于小于 1000 且大于 0.0001 的小数，则整数部分会按数据的原格式显示，而小数只会显示到小数点后面 4 位，如 999.01236 即显示 999.0124；对于大于 1000 或小于 0.0001 的小数，其显示格式与 format short e 相同，即小数部分只显示 5 位数，如输入 0.00001 便会显示 1.0000e-005，而输入 1000.002345 则会显示 1.0000e+003，而对于整数，绝对值小于 10^9 则按原样输出，而大于等于 10^9 的数则与 format shore e 相同。可以通过实验来比较它与 format short e 之间的联系与区别。

(3) format long 长格式，显示 15 位数。即将所有的小数都用 e 格式输出，e 左边为 15 位数，如输入 1.2 则会输出 1.200000000000000e+000，这与 format long e 格式基本相同；而对于整数，绝对值小于 10^9 的数按原样输出，而大于等于 10^9 的数则按 e 格式输出，如输入 10000 则显示 10000，而输入-1000000000 则显示 -1.000000000000000e+009。

(4) format shore e 短格式 e 方式，对于任意小数采用 e 格式，只显示 5 位小数，如输入 1.2 则会显示 1.2000e+000；对于整数，其显示格式与用 format short 格式时相同。

(5) format long e 长格式 e 方式，显示 15 位小数。对于任意小数都会采用 e 格式显示，如输入 1.2 则显示 1.200000000000000e+000；对于整数，其显示格式参照 format short。

(6) format short g 最优化短格式，最大显示 5 位数据，系统会根据数据的大小及形式采用比较好的显示数据的方式，如略去小数后面的零等。如输入 1.234e1 则显示 12.34，输

入 12345.67890 则显示 12346，输入 1234567.8988 则显示 1.2346e+006，输入 0.02000 则显示 0.02 等，其显示方式看起来比较灵活和更加合理。对于整数的显示方式，可以参照 format short 格式。

(7) foramt long g　最优化长格式，最大显示 15 位数据。其显示规则大体与 format short g 相同，现举几个例子来加以说明。输入 1.2 则显示 1.2，输入 1.23456789000 则显示 1.23456789，输入 1.2233445567898765423456 则显示 1.22334455678988，输入 1234567898765432.2345 则显示 1.23456789876543e+015。

(8) format hex　十六进制格式。显示二进制双精度数的 16 进制形式，如输入 1.2 则显示 3ff3333333333333，输入-9.003 则显示 c0220189374bc6a8。

(9) format bank　货币银行格式。保留小数点后两位小数，对于整数亦是如此。但并不是简单的四舍五入法，如输入 0.355 则显示 0.36，而输入 0.345 则显示 0.34，对于任意整数其后都要显示两位小数，如输入 1 则显示 1.00。

(10) format　rat　有理格式。对于整数则照常显示，而对于小数则用两个整数相除的方式显示。如输入 0.5 及 1/2 则都会显示 1/2 ，输入 3.66667 则显示 11/3 。注意：用有理小数表示分数是有一定精度要求的，如输入 3.6667 则只会显示 36667/10000 而不会显示 11/3。

(11) format+　紧密格式。用“＋”、“－”或空格表示数据为正、负、或零。如输入 2－3 则会显示－号，输入 1 会显示＋号，输入 0 则会输出一个空格。

(12) format compact　紧凑格式，即在输入的命令及回显结果之间不加空行。

(13) format loose　疏松格式，即在输入命令及回显结果之间都加空行。

下面举例来说明以上两个命令之间的区别。

例 10　format compact

```
a=[2;3]
a =
     2
     3
format loose
a
a =
     2
              3
```

说明：不论采用什么样的显示格式，数据在内存中的格式是不会变的，即不影响数据的存储。对于 a=1/3，当用 format short 格式显示时是 0.3333，而改用 format rat 显示时仍是 1/3，而不会因为上面已经用 short 格式显示过而丢失精度。

2.3　关系运算符

关系运算符主要用来对数与矩阵、矩阵与矩阵进行比较，并返回反映二者之间大小关系的由数 0 和 1 组成的矩阵。基本的关系运算符主要有：>、<、<=、>=、==、~=6 个。其

用法分别为：

1. 等于："=="

A==B，如果 A 和 B 都为矩阵，则 A 和 B 必须具有相同的维数，运算时将 A 中的元素和 B 中的对应元素进行比较，如果两者相等，则在输出矩阵的对应位置输出 1，反之输出 0。如果 A 和 B 有一个为数，则将这个数与另一个矩阵的所有元素进行比较。无论何种情况，返回结果都是与参与运算的矩阵有相同维数的由 0 和 1 组成的矩阵。其余关系运算中对 A 和 B 的要求和返回结果的维数所满足的条件亦是如此。

函数 eq(A，B)亦是对两个对象进行比较，看是否相等。其中，A 和 B 可以是矩阵和数值，但也可以是其他的对象，如 figure 对象。

例 11　a=[1 2；2 1]

```
b=[1 1; 2 2];
eq(a,b)
ans =
      1           0
      1           0
g=a==1        %求 a 中等于 1 的元素个数
g =
      1           0
      0           1

sum(sum(g))
ans =
      2
```

2. 不等于："~="

A~=B，与 A==B 相反，如果 A 和 B 中相同位置上的对应元素不相等，则在返回矩阵的相应位置上输出 1，反之输出 0。由此可见，所得矩阵应该与 A==B 得到的矩阵中 0 和 1 的位置相互颠倒。

同样，函数 ne(A，B)对两个对象 A 和 B 进行比较，也可以用于对两个矩阵进行比较。

例 12　a~=b

```
ans =
      0           1
      0           1
```

3. 小于："<"

A<B，如果 A 矩阵中的元素小于 B 矩阵中相应位置的元素，则在输出矩阵的此位置上输出 1，反之则输出 0；如果其中之一为数值，则将这个数与另一对象的每一个元素进行比较。

函数 lt(A，B)亦是判断 A 是否小于 B，A 和 B 可以是矩阵、数值或任意其他的对象。

例 13　a<2

```
ans =
      1         0
      0         1
```

4. 大于：">"

A>B，如果 A 矩阵中的元素大于 B 矩阵中相应位置的元素，则在输出矩阵的此位置上输出 1，反之则输出 0；如果其中之一为数值，则将这个数与另一对象的每一个元素进行比较。

函数 gt(A，B)亦是判断 A 是否大于 B，A 和 B 可以是矩阵、数值或任意其他的对象。

例 14　a>1

```
ans =
      0         1
      1         0
```

5. 小于等于："<="

A<=B，如果 A 矩阵中的元素小于或等于 B 矩阵中相应位置的元素，则在输出矩阵的此位置上输出 1，反之则输出 0；如果其中之一为数值，则将这个数与另一对象的每一个元素进行比较。

函数 le(A，B)亦是判断 A 是否小于或等于 B，A 和 B 可以是矩阵、数值或任意其他的对象。

例 15　a<=1

```
ans =
      1         0
      0         1
```

6. 大于等于：">="

其运算方法同上，在此不再赘述。

2.4　逻辑运算及逻辑函数

逻辑运算和逻辑函数在计算机语言中是普遍存在的，在 MatLab 中包含与、或、非、异或 4 种基本的逻辑运算。逻辑表达式和逻辑函数的值应该为一个逻辑量"真"或"假"。MatLab 系统在给出逻辑运算的结果时，以数值"1"代表逻辑"真"，以"0"代表"假"，但在判断一个量是否为"真"时，以 0 代表"假"，以任意的非零值代表"真"。MatLab 的逻辑运算也是以矩阵为基本运算单元的。

2.4.1　逻辑运算

符号“&”、“|”、“~”、“xor”分别代表逻辑运算中的与、或、非、异或，它们作用于数组元素。0 的逻辑量为“假”，而任意非零数的逻辑量为“真”。逻辑运算的运算法则有：

- 在逻辑数组中，0 代表逻辑“假”，1 代表逻辑“真”。
- 如果两个标量 a 和 b 参加运算，则各逻辑运算的运算规则如表 2.1 所示：

表 2.1　逻辑运算规则

输入		“与”	“或”	“异或”	“非”
A	B	a&b	a|b	xor(a,b)	~a
0	0	0	0	0	1
0	1	0	1	1	1
1	0	0	1	1	0
1	1	1	1	0	0

- 如果两个维数相同的矩阵参与运算，则将 A 和 B 相同位置上的元素按标量逻辑运算的规则进行运算，结果返回与矩阵 A 和 B 同样大小的矩阵，其元素由同位置上的 A 和 B 的元素进行逻辑运算的结果所决定。
- 如果标量 a 和矩阵 A 参加运算，则将 a 和 A 中的所有元素进行逻辑运算，返回结果是由 0 和 1 组成的与 A 具有同样维数的矩阵。
- 在逻辑操作符、关系操作符和计算操作符 3 者中，逻辑操作符的优先级最小，但是逻辑“非”的优先级最高。
- 在逻辑“与”、“或”、“非”3 者中，“与”与“或”有相同的优先级，从左到右依次执行，而都低于“非”的优先级。
- 通过增加“()”可以改变各操作符之间的优先级。

下面通过例 16 理解各个逻辑操作的运算法则。

例 16

```
a =[1 2 ; 2  1];                    %输入矩阵 a
b=[0 2;3  0];                       %输入矩阵 b
a&b                                 %求逻辑“与”
ans =                               %结果显示
     0          1
     1          0
a|b                                 %求逻辑“或”
ans =                               %结果显示
     1          1
     1          1
~b                                  %求逻辑“非”
ans =                               %结果显示
     1          0
     0          1
```

```
xor(a,b)                          %求逻辑“异或”
ans =                             %结果显示
     1         0
     0         1
```

通过例 17 着重理解各操作符的优先级。

例 17

```
~a-1>=1                           %逻辑表达式
ans =                             %结果显示
     0         0
     0         0
a-1>=1&0                          %逻辑表达式
ans =                             %结果显示
     0         0
     0         0

~(a-1>=1)                         %逻辑表达式
ans =                             %结果显示
     1         0
     0         1
~a-1                              %逻辑表达式
ans =                             %结果显示
    -1        -1
    -1        -1
1 | 0 & 0                         %逻辑表达式
ans = 0                           %结果显示
0 & 0 | 1                         %逻辑表达式
ans = 1                           %结果显示
```

在 M 文件中，可以用 and(A，B)，or(A，B)，not(A)分别进行“与”、“或”、“非”的操作。亦可以在命令行中进行直接输入，其结果与用“&”、“|”、“~”运算所得结果相同。

```
and(a,b)
ans =
     0         1
     1         0
```

2.4.2 逻辑函数

MatLab 从 3.0 版本开始提供逻辑函数，这些函数在交互运算及进行矩阵的变化中非常有用，可以很方便地查找或替换矩阵中满足一定条件的部分或所有元素，在使用过程中应认真体会每个函数的具体用法，才能在实际应用过程中灵活运用。

1. all：判断是否所有元素为非零数

如果要获得矩阵或向量中非零元素的位置或个数，可以利用 all 函数。此函数对于向量

a 或矩阵 A 有 all(a)，all(A)，all(A，dim) 3 种调用方式。对于向量 a(行向量或列向量)，如果向量中的每个元素都是非零数，all(a)返回逻辑“真”，即“1”，如果至少一个元素为零，则返回逻辑“非”，即“0”。对于矩阵 A，all(A)是作用于列向量上的，即如果矩阵 A 的某列所有元素都是非零数，则返回结果的当前列为逻辑“真”，即为“1”；如果至少有一个为零，则返回结果的当前列为逻辑“假”，即“0”。显然，返回结果为与矩阵 A 具有同列维的行向量。如果 A 是多维矩阵，则 all(A)将第 1 个不是单维的维作为向量，进行运算，运算规则同向量运算。all(A，dim)将指定的第 dim 维作为向量进行运算。下面举些例子来说明函数 all()的用法。

例 18　判断矩阵 A=[1　2　3；0　2　5]的所有元素是否都大于或等于 1

```
all(all(A>=1))
ans =  0
A>=1
```

得到矩阵：

```
1     1     1
0     1     1
all(A>=1)
```

得到向量：

```
0     1     1
```

而对此向量再作一次 all 运算，便可以得到最终的结果 0。

从以上分析不难看出，如果 A=[1　2　3；6　2　5]，则 all(all(A>=1))=1。

从例 19 中仔细体会 all()函数运算的规则。

例 19

```
A=[1  2  3;0  2  5]                    %输入矩阵
all(A)                                 %求逻辑运算
ans =                                  %结果显示
    0     1     1
all(A,2)                               %逻辑运算
ans =                                  %结果显示
    1
    0
d=rand(1, 2, 3)                        %生成 1×2×3 维随机矩阵
d(:,:,1) = 0.3704    0.7027            %结果显示
d(:,:,2) = 0.5466    0.4449
d(:,:,3) =0.6946    0.6213
all(d)                                 %逻辑运算
ans(:,:,1) = 1                         %结果显示
ans(:,:,2) = 1
ans(:,:,3) = 1
```

从例 19 可以看出，三维列向量 d 进行 all()的逻辑运算时，是将第二维作为向量进行计算的，而不是将第一维作为一个向量。

在 M 函数中，可以充分利用 all()函数的功能，对向量或矩阵进行判断，并根据返回结果作出相应的反映。如：

```
if all(all(A>0.5))
    %将要执行的代码放到此处
end
```

2. any：判断是否有一个向量元素为非零

在矩阵处理中，有时候会需要判断矩阵中的元素是否有零或非零值，如在对矩阵进行数组除时，就需要判断作除数的矩阵是否有零元素，any 函数可以实现这一功能。此函数也有 4 种调用格式：any(a)，any(A)，any(A，dim)。对于向量 a(行向量或列向量)，如果向量中至少有一个元素为非零数，any(a)返回逻辑“真”(即“1”)，而如果所有元素都为零，则返回逻辑“非”(即“0”)。对于矩阵 A，any(A)与 all(A)一样，也是作用于列向量上的，即如果矩阵 A 的某列中存在某个元素为非零数，则返回结果的当前列为逻辑“真”，即为“1”，如果所有元素都为零，则返回结果的当前列为逻辑“假”，即“0”。如果 A 是多维矩阵，则 any(A)将第 1 个不是单维的维作为向量，进行运算，运算规则同向量运算。any(A，dim)将指定的第 dim 维作为向量进行运算。下面举例加以说明：

例 20

```
A =[0      2      3;  0      2      5]          %输入矩阵
any(A)                                          %逻辑运算
ans =                                           %结果显示
     0      1      1
any(A,2)                                        %逻辑运算
ans =                                           %结果显示
     1
     1
```

any()函数的其余规则与 all()基本相同，在此不再赘述。

3. exist：查看变量或函数是否存在

在 MatLab 程序设计中，有时候需要知道变量是否已经被定义过，即是否存在于当前内存中，有时候还需要详细了解变量的类型，这时 exist 函数就显得非常有用。a=exist('A')返回变量或函数的状态或类型，a 的值及 A 对应的状态或类型分别为：

a 的值　A 的状态或类型

0　如果对象 A 不存在或没在 MatLab 的搜索路径下

1　如果 A 是工作空间中的一个变量

2　如果 A 是一个 M 文件或是一个在 MatLab 搜索路径下的未知类型的文件

3　如果 A 是一个 MatLab 搜索路径下的 MEX 文件

4　如果 A 是一个 MatLab 搜索路径下的已编译的 SIMULINK 函数(MDL 文件)

5　如果 A 是 MatLab 的内置函数

6　如果 A 是一个 MatLab 搜索路径下的 P 文件

7　如果 A 是一个路径，不一定是 MatLab 的搜索路径

如果对象 A 存在于 MatLab 的搜索路径下，但并不是 MatLab 可以识别的非 M 文件，即非 MDL 文件、P 文件、MEX 文件时，exist('A')或 exist('A.ext')将返回 2。

注意：exist('A')中，A 可以是当前路径下的子目录或相对路径。

例 21　MatLab 的当前路径为 D:\MatLab，则输入 exist('bin')便会得到

```
ans =   7
```

当然也可以检验计算机上的任意一条路径，但必须要输入全部路径。如：

```
exist('c:\windows')
ans =  7
filtdes.p 是 toolbox\signal 目录下的一个 P 文件，输入：
exist('filtdes')
```

得到

```
ans =   6
exist('sin')                %sin 为 MatLab 的内置函数
ans =  5
```

flag=exist('A', kind)，如果 MatLab 找到指定类型 kind 下的对象 A 的话，此语句返回逻辑“真”，即 flag 值为 1，否则返回逻辑“假”，flag 值为 0。其中，kind 参数的取值可以为

- exist('A', 'var')　　仅检查工作空间中的变量。
- exist ('A', 'builtin')　　仅检查 MatLab 的内置函数。
- exist ('A', 'file')　　检查 MatLab 搜索路径下的文件和路径。
- exist ('A', 'dir')　　仅检查路径。

下面举例说明。

例 22　exist('a', 'var')　%在工作空间中变量中查找

ans =　1　　　%表明找到名为“a”的变量

exist('a', 'builtin')　　%在内置函数中查找

ans =　0　　　%不存在名为“a”的内置函数

4. find：找出向量或矩阵中非零元素的位置标识

在许多情况下，都需要对矩阵中符合某一特定条件的元素的位置进行定位，如将某一矩阵中为零的元素设为 1 等。如果这个矩阵的元素非常多，手工修改就非常麻烦，而灵活运用 find 函数和各种逻辑及关系运算可以实现绝大多数条件的元素定位。find 函数的基本用法有 k=find(A)，[i，j] = find(A)，[i，j，v] = find(A)，这是个很有用的逻辑函数，在对数组元素进行查找、替换和修改变化等操作中占有非常重要的地位，熟练运用可以方便而灵活地对数组进行操作。

k=find(A)　此函数返回由矩阵 A 的所有非零元素的位置标识组成的向量。如果没有非零元素则会返回空值。

例 23

```
a=[1  0;3  1; 0  2]
```

```
find(a)                %查找 a 中的非零元素
ans =                  %结果显示
    1                  %可见元素的标识是按列进行的，即从 1 开始，数完第
    2                  %1 列再数第 2 列，依次数下去
    5
    6
b(:,:,1) = 0         0
        0.2311    0.4860
b(:,:,2) = 0.8913    0.4565
        0.7621    0.0185
find(b)
ans =    2    4    5    6    7    8
```

从例 23 中不难发现，多维数组的元素标识是从低维到高维依次进位的，如对于一个三维数组 B，它的元素 B(1，1，1)，B(2，1，1)，B(1，2，1)，B(2，2，1)，B(1，1，2)，B(2，1，2)，B(1，2，2)，B(2，2，2)的标识依次为 1，2，3，4，5，6，7，8。

[i，j] = find(A) 此函数返回矩阵 A 的非零元素的行和列的标识，其中 i 代表行标而 j 代表列标。此函数经常用在稀疏矩阵中。在多维矩阵中通常将第一维用 i 表示，将其余各维作为第二维，用 j 表示。如对于上面的三维矩阵 b，有：

例 24 [i，j]=find(b)

```
i =  2    2    1    2    1    2
j =  1    2    3    3    4    4
```

[i，j，v] = find(A) 此函数返回矩阵 A 的非零元素的行和列的标识，其中 i 代表行标而 j 代表列标，同时，将相应的非零元素的值放于列向量 v 中，即 i 和 j 的值与[i, j] = find(A)取值相同，只是增加了非零元素的值这一项。

例 25 [i，j，k]=find(b)

```
i =  2    2    1    2    1    2
j =  1    2    3    3    4    4
k =   0.2311   0.4860   0.8913   0.7621   0.4565   0.0185
```

下面举例说明此函数的重要作用。

例 26 找出 a 中小于−3 的元素的位置

```
find(a<-3)
ans = []
```

例 27 找出 a 中等于 0 的元素的位置

```
find(a==0)
ans = 3
      4
```

例 28 将矩阵 a 中等于 1 的元素换为-1

```
a(find(a==1))=-1
a =
```

```
-1    0
 3   -1
 0    2
```

例 29　将矩阵 a 中等于－1 的元素的值换成矩阵 c 中相应位置上的元素。即部分矩阵元素值的替换

```
c =[ 3    0;2    5; 8    -3]
a(find(a==-1))=c(find(a==-1))
a =  3    0            %可见，上面的两个－1 已分别被替换为 3 和 5
     3    5
     0    2
```

例 30　将矩阵 a 中等于 0 的元素删除。

```
a(find(a==0))=[]
a = 3    3    5    2
```

需要说明的是，在矩阵计算中通常可以采用这种方法进行矩阵删除，即将矩阵的某个或某行元素直接赋值为零。

如删除 c 矩阵的第 3 行，可以这样做：

```
c(3,:)=[]
c =  3    0
      2    5
```

上面“( )”中“：”为操作符，代表第 3 行的所有元素。

5. finite：确认矩阵元素是否为有限值

finite(A)　如果矩阵 A 中的元素为有限值，则此函数在返回矩阵的相应位置上输出 1，否则输出 0。有限值为具有确定值的数，而 NaN，＋Inf，－Inf 等都被视为无限值。函数 isinf(A) 判断矩阵 A 的元素是否为无限值，用法与 finite 相同。

说明：NaN 称为不确定值，通常由 0/0，Inf±Inf，Inf/Inf 及 NaN 与其他任何数进行运算得到；Inf 称为无穷大数，可以由任意非零实数除以零得到，而复数除以零会得到 NaN 数。

例 31　finite(c)

```
    ans =
    1    1
    1    1
    d=[1  +inf; nan -inf]
finite(d)
ans =
    1    0
0
```

例 32　isinf(d)

```
ans =  0     1
       0     1
```

6. isempty：确认矩阵是否为空矩阵

不要把空矩阵、零矩阵及矩阵不存在等 3 个概念混淆，空矩阵说明矩阵存在，但是矩阵没有元素；零矩阵是指矩阵的所有元素都为零；矩阵不存在是指当前的工作空间中没有定义此矩阵变量。isempty(A)可以判断一个存在的矩阵变量 A 是否为空矩阵，如果 A 矩阵为空矩阵则返回逻辑“真”，否则返回逻辑“假”，一个零矩阵至少有一维是零，如 0×0，0×5，0×3×3 等。零矩阵没有任何元素，可以用函数 size(A)来判断，如果其中有一维为零，则 A 就是零矩阵。

例 33

```
a = []
size(a)
ans = 0     0
a=rand(3,3,3);
a(:,:,:)=[]
a =Empty array: 0-by-3-by-3
size(a)
ans = 0     3     3
```

说明 a 矩阵是个 0×3×3 维矩阵，因而 a 是个零矩阵。

```
isempty(a)
ans = 1
```

7. isequal：判断几个对象是否相等

isequal(A，B，C…) 如果要判断的所有对象 A，B，C…具有相同的类型、大小和内容，对于矩阵来说，就是所有矩阵的维数相同，而且矩阵元素的数值相同，如果满足这样的条件，此函数返回逻辑“真”，反之，只要有一个对象与其他对象不相同，就会返回逻辑“假”。

例 34

```
a =[0.1509    0.6979]
b =[0.1509    0.6979]
c='hello world'
d =[0.1509;0.6979]
isequal(a,d)
ans = 0
isequal(a,b)
ans = 1
isequal(a,b,c)
ans =  0
```

8. Isnumeric：判断对象是否是数据

Isnumeric(A)　如果 A 是数据矩阵，如系数矩阵、双精度矩阵、复数矩阵等，此函数返回逻辑“真”，反之，如果 A 是字符串、结构体矩阵等，则返回逻辑“假”。

例 35

```
isnumeric(a)               %a 为实数矩阵
ans =  1
e=[1+2*i 3*i]
isnumeric(e)               %e 为复数矩阵
ans =  1
isnumeric(c)               %c 为字符串
ans =   0
```

还有一些逻辑函数，也比较常用，为了保持完整性，将其部分列出，以供参考。

issparse　　判断是否为稀疏矩阵
isstr　　判断是否为字符串
islogical　　判断一个矩阵是否为逻辑矩阵
isfield　　判断对象是否为某个结构体矩阵地域
isstruct　　判断是否为结构体
ishandle　　判断是否为图像句柄

2.5　字符串操作

在许多的计算机高级语言中，字符串处理一向是作为一个非常重要的部分。由于 MatLab 注重矩阵的计算和处理，因而字符串在 MatLab 中的地位没有在其他高级语言中那么举足轻重，但 MatLab 处理字符串的功能还是非常强大的，它提供了完善的处理字符串的函数，如对字符串进行比较、取子串等。同样，MatLab 对字符串的操作也是建立在矩阵处理的基础上的，可以在 2.5 节的学习中仔细体会。

2.5.1　MatLab 中的字符串符号

在 MatLab 中，要建立一个字符串变量，可以这样建立：S='字符串'，即用' '将输入的字符串括起来。注意：不是" "，这与一些其他的高级语言不同。而要建立一个字符串矩阵，则可以这样输入：

```
SA=['string11'  'string12'  …
'string21  'string22'  …
'stringn1' 'stringn2' …]
```

与数组不同，字符串矩阵的每一行字符串元素的个数可以不同，但是每一行的所有字符串中字符的总个数必须相同，如果不满足这个条件，即使每行中字符串的个数相同，也会出错。事实上，MatLab 将一个行内的所有字符串都合并起来，构成一个字符串，单个字

符串之间不加空格，这正是每行中输入的字符串的个数可以不相同的根本原因。

例 36

```
SA=['hello' 'good' 'student'
'become' 'my' 'friend' 'Ok']
SA =hellogoodstudent
BecomemyfriendOk
```

利用这个特点，可以用[ ]将任意字符串连接起来。

例 37 将例 36 中 SA 的上下两行连接起来，可以这样操作：

```
 [SA(1,:)  SA(2,:)]
ans =hellogoodstudentbecomemyfriendOk
```

S='任意字符串'，是一个由字符的 ASCII 码组成的向量，而实际所显示的是由给定的字体经过编码后的字符而不是一些数码，变量 S 的长度便是字符串中的字符的个数。由于 MatLab 中' '是标识字符串的特殊字符，因而要在字符串中输入' '必须通过 2 个' '来表示，而 "" 可以直接输入。

例 38

```
s='hello world'
size(s)
ans =
    1    11
    c='It''s a dog'          %此中字符串内为两个' '，而不是" "
c =                          %显示结果
It's a dog
d='She said:"I am OK!"'      %字符串内是两个""，外面是两个' '
d =
She said:"I am OK!"
```

说明：MatLab 在处理字符串矩阵时是把它当作数据矩阵来处理的，字符串的每个字符都是矩阵的每个元素，这样字符串矩阵也应当满足数据矩阵的所有条件，即要求每行的元素个数必须相同，上下两行的字符总数必须相同。

字符串及字符串矩阵可以进行加、减、乘、除四则运算和其他的数学运算。由于 MatLab 是将字符串及字符串矩阵当作数据矩阵来处理的，因而在进行这些运算时，实际上是由字符串的各个字符的 ASCII 码组成的数据矩阵之间的数学运算。利用下面将要给出的字符串操作函数不难检验出这一点。通常可以打印的字符的 ASCII 码是在 32～127 范围之间，但任何 8 位二进制数都是合法的，范围在 0～255 之间，如果数值不是正整数，或是超出了上面的范围，则实际上是打印出 ASCII 码为 fix(rem(A，256))的字符。

例 39

```
'e'+'f'
ans =203
e'*'f'
ans =10302
```

其中

```
abs('e')=101, abs('f') =102
sin('e')
ans =0.4520
c='bbb'-'aaa'
c = 1     1     1
size(c)
ans = 1     3
```

而下面的操作便会出错：

```
c='bb'-'aaa'
??? Array dimensions must match for binary array op.
```

因为两个字符串转换成矩阵后并不满足矩阵运算的条件。

2.5.2 一般通用字符串操作

通用字符串操作包括字符串与 ASCII 间的转换、字符串与数据间的相互转换，字符串大小写间的转换、字符串中空格的删除等，这些操作都是对字符串最常用和最基本的操作。在其他高级语言中的字符串操作部分一般都含有这些操作，因而可以称之为通用字符串操作。下面详细介绍这些操作。

1. 将整数数组转换为字符串

S=string(A)　其中 A 为正整数数组，这个函数的作用是将一个整数数组转换成字符串矩阵，字符串中字符的 ASCII 码即是 A 中相应元素的值。

例 40

```
string(b)                          %将数组 b 转换成相应的字符串矩阵
ans =
hello
world
```

例 41　将任意的正整数矩阵转换为其相应的字符串矩阵

```
    h =[1     2     3     4     5     6     7
       101    23    44    56   ·65    67    89]
string(h)
ans =□□□□□□□                       %方框表示此字符并不能在屏幕上很好地显示出来
     e□,8ACY
```

2. 将 ASCII 码转换为字符串

char(A)　此函数将由正整数组成的矩阵 A 转换成字符串矩阵，矩阵 A 的元素一般要在 0~65 535 之间，超出这个范围的是没有定义的，但也可以显示出结果，只是系统会给出超出范围的警告。通常的处理便是将这个数对 65 536 取模，变成 0~65 535 之间的数再进行转换。

例 42

```
char(h)
ans =

e:.,8ACY                          %此结果与 sring(h)得到的一样
    char(66666)
Warning: Out of range or non-integer values truncated during conversion
from
double to character.
ans =j
char(rem(66666,65536))        %rem(66666,65536)为将 66666 对 65536 求余
ans =j
```

例 43　打印出 ASCII 码从 32～127 所有可以显示的字符

```
    char(reshape(32:127,48,2)')
ans =
 !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNO
PQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~
```

S=char(C)　如果 C 是由字符串组成的单元阵，此函数将单元阵 C 转换成字符串矩阵，字符串矩阵的每行就是单元阵的每个元素，且用空格将每个字符串补齐，以保证字符串矩阵的合法性。也可以用 cellstr()函数将一个字符串矩阵转换为一个字符串单元阵。

例 44

```
g ={ 'hello'   'my'    'country'}
k=char(g)
k =
'hello  '                     %为了便于理解，在此加上' '，实际输出%没有引号
'my     '                     %可见 k 的每行字符串都用空格补成长度相同
'country'
cellstr(k)                    %将字符串矩阵转换为单元矩阵
ans = 'hello'
    'my'
    'country'
```

S=char(s1，s2，s3，…)　此函数以各个字符串 s1，s2，s3，…为每行构成字符串矩阵 S，并自动以适当的空格追加在较短的字符串的后面，使各行的字符串的字符个数相同，以构造合法的字符串矩阵。参数中的空字符串也会被空格填充为相同大小的空格字符串。

例 45

```
S=char('this','is','a','','bed')
S =
'this'                        %为了便于理解，在此加上' '
'is  '                        %字符串末尾补充 2 个空格
'a   '                        %字符串末尾补充 3 个空格
'    '                        %此行是由 4 个空格补充空串得到
'bed '                        %字符串末尾补充 1 个空格
```

3. 将字符串转换成ASCII码

abs(S) S为字符串，此函数返回S的每个字符的ASCII码，结果是一个整数矩阵，可以当作一般的矩阵处理。

例 46

```
e=['hello';'world']
b=abs(e)
b =
   104   101   108   108   111
   119   111   114   108   100
```

4. 将字符串转换为相应的ASCII码

double(S) 此函数的作用与 abs(S)有相同之处，它是将符号矩阵或字符串转换成由双精度型的浮点数组成的矩阵。在符号运算中，它是按双精度形式计算一个符号表达式的结果，具体方法将在第3章中详细讨论。在此只举几个例子来说明：

例 47

```
double('2+3')                  %将字符串转换成ASCII码
ans = 50    43    51
w=sym('2+3')                   %将字符串转换成符号表达式
w =2+3
double(w)                      %以双精度形式计算符号表达式的值
ans = 5
```

5. 确认是否为字符串

isstr(S) 这实际上是个逻辑函数，如果S是字符串或是字符串矩阵，此函数返回逻辑"真"；否则，如S是字符串组成的单元阵、数组等，就返回逻辑"假"。

例 48

```
a =[hello; world]
isstr(a) = 1
isstr('a') = 1
g={'hello' 'my' 'country'}
isstr(g) = 0
```

6. 删除字符串结尾处的空格

S=deblank(C)，该命令表示如果 C 是字符串，则将字符串后面的所有空格都删除，字符串内部的空格保留。如果C是字符串矩阵，在删除每行的字符串后面的空格时，首先要保证删除空格后所有行的字符总数必须相等，因此，并不是简单地将各行字符串后面的空格删除，而是在保证上述条件的基础上删除最大空格数。如果C是单元阵，则函数将单元阵中每个字符串元素后面的空格都删除，不必保证每个元素的字符总数必须相同，这与

对矩阵进行操作时有所不同。

例 49

```
c='aaa   '
size(c)=1     6
deblank(c)='aaa'
a=['he  llo    '              %空格数分别为 2 和 4
   'wor   ld   ']             %空格数分别为 3 和 3
size(a) = 2    11
b=deblank(a)
b =he  llo                    %空格数分别为 2 和 1
  wor   ld                    %空格数分别为 3 和 0
size(b) =2     8
```

7. 输入空格符

用于输出 n 个空格数。此函数在调整输出格式、要输出多个空格时很有用，可以精确地输出需要的空格数。通常与 disp()函数联用，对输出格式进行调整。

例 50

```
s=blanks(5)
size(s) = 1     5
f=['20 spaces' blanks(20) 'end']
f =20 spaces                    end
```

8. 将字符串变成字符串矩阵

S=str2mat(s1，s2，s3，…) 此函数与 S=char(s1，s2，s3，…)命令的作用完全相同，生成矩阵的格式也完全相同，在此不再赘述。

9. 将字符串转换成字符串矩阵

S=strvcat(s1，s2，s3，…) 此函数与字符串矩阵生成函数用法基本相同，不同的是在生成矩阵时会将其中的空行忽略掉，而不是将其用空格补起。

例 51

```
S=strevcat('this','is','a','','bed')
S =
'this'
'is  '
'a   '                        %空行已经被忽略掉
'bed '
```

10. 将字符串进行大小写转换

upper(S) 函数将字符串或字符串矩阵 S 中的所有的小写字母转换成大写，原有的大写字母保持不变。

Lower(S) 函数将字符串或字符串矩阵 S 中的所有的大写字母转换成小写，原有的小写字母保持不变。

例 52

```
s=['hello';'WORLD']
upper(s)
ans =HELLO
 WORLD
lower(s)
ans =hello
 world
```

11. 将字符串作为命令执行

a=eval('字符串表达式') 此函数返回由字符串表达式执行的结果。可以将各个不同部分放在“[ ]”内以形成一整条命令。这个函数在 M 文件中进行交互式执行命令时很有用。

例 53

```
eval('a=[2:4;6:8]')
a =2     3     4
   6     7     8
```

例 54 要依次对 a1～a9 分别赋值 1～9，可以这样操作：

```
for i=1:10
eval(['a' abs('0')+i '=' abs('0')+i])
end
a1 =1                       a6 =6
a2 =2                       a7 =7
a3 =3                       a8 =8
a4 =4                       a9 =9
a5 =5
```

例 55 如果要载入几十个甚至上百个文件，用手工操作十分繁琐，然而灵活运用 eval 函数可以自动完成这一工作。假设函数名从 a1.m～a100.m，操作如下：

```
for i=1:100
eval(['load(''a' int2str(i) '.m'')'])
end
```

此语句便会自动将 100 个文件装载入内存。

[a1，a2，a3，...] = eval('字符串表达式') 如果表达式有多个返回结果时使用这个函数。

这条语句等价于 eval('[a1, a2, a3, …] =字符串表达式')，计算结果亦相同。

例 56 执行以下两条语句，可以得到相同的结果：

```
a1 =[1    2; 3    4]
[l u]=eval('lu(a1)')                    %lu()为三角分解函数，可以有两个返回
                                        %结果
eval('[l u]=lu(a1)')
l =
    0.3333    1.0000
    1.0000         0
u =
    3.0000    4.0000
         0    0.6667
```

eval('表达式 1', '表达式 2') 提供捕捉错误的功能。此函数首先执行"表达式 1"，如果执行成功，就返回执行结果，如果在执行"表达式 1"期间出现错误，函数便跳过"表达式 1"而去执行"表达式 2"。在 M 文件或宏命令中，可以很容易捕捉到错误而不至于因为一个错误而导致程序停止运行。如果想了解到底发生了什么错误，可以通过 lasterr 命令来查看错误代码。

例 57

```
eval(['load(''a' int2str(2) '.m'')'],'disp(''现在开始执行出错命令'')')
%上面语句要求装载 a2.m 文件
%现在开始执行出错命令                      %eval 函数执行"表达式 2"的命令
lasterr
ans =
Error using ==> load                     %因为 a2.m 文件不存在，因而导致装载文
a2.m: Can't open file.                   %件出错
```

例 58 可以利用 eval()函数在 M 文件中构造选择性菜单

建立名为 mydemo.m 的文件，内容如下：

```
Dsp = ['ardemo ' ;  'quaddemo' ;   'cplxdemo' ;   'fitdemo '];
                                         %注意：上面每个字符串的字符个数相等
disp(Dsp)                                %显示各命令
n = input('请选择一条命令(1~4) : ');     %读入整数 n
eval(Dsp(n,:),'disp(''你选择的命令不能被执行，请重新选择'')');
                                         %执行相应的命令
```

在命令窗口下输入mydemo即可以执行。

2.5.3 字符串比较操作

字符串比较操作主要涉及对字符串按字母顺序进行比较及对字符串进行匹配、查找、替换和提取子串等一系列的操作。这些是比较有用的操作，也可以归入通用命令部分，为了便于学习，将其单独列出。

1. 两个字符串比较

strcmp('string1', 'string2') 将两字符串进行比较，如果两字符串相等，此函数返回逻辑“真”，否则返回逻辑“假”，即，此函数只能判断两字符串是否相等，而不能判断按字母顺序谁在谁的前面。

例 59

```
strcmp('good','good')
ans = 1
strcmp('good','bad')
ans =  0
```

strcmp(C1, C2) 如果 C1 和 C2 都是由字符串组成的大小相同的单元阵，此函数返回一个与单元阵相同大小的逻辑矩阵。如果单元阵 C1 和 C2 相同位置上的字符串相同，则在逻辑矩阵的相应位置上输出 1，否则输出 0。C1 和 C2 其中之一或全部可以为字符串或字符串矩阵，但返回的逻辑矩阵与单元阵有相同的大小。

例 60

```
c1={'yes' 'no'; 'MatLab'   'MatLab'}
c2={'yes'    'NO';'MatLab'  'MatLab'}
c3='yes'
c5=c4=['yes' 'no'; 'mmm' 'nn']
strcmp(c1,c2)
ans =1     0
     0     1
strcmp(c1,c3)
ans =1     0
     0     0
strcmp(c5,c4)
ans = 1
```

需要说明的是，前导或后导空格也会参与比较，比较函数对大小写是敏感的。

2. 比较字符串的前 n 个字符

strncmp('string1', 'string2', n) 如果 2 个字符串的前 n 个字符相同，则此函数返回逻辑“真”，否则返回逻辑“假”，比较函数对大小写敏感。

例 61

```
s1='MatLab', s2='MatLab'
strncmp(s1,s2,3)=1, strncmp(s1,s2,4)=0
```

strncmp(C1, C2, n) 如果 C1 和 C2 为由字符串组成的大小相同的单位阵，则此函数将相同位置的字符串的前 n 个字符进行比较。如果相同就在相同的位置输出 1，否则输出 0；如其中之一为字符串，则将单位阵中的所有字符串都与这个字符串进行比较，返回与单位阵相同大小的逻辑阵。

例 62

```
s='MatLab'
strncmp(c1,s,4)=  0     0
                  0     0
strncmp(c1,s,3)= 0     0
                 1     0
```

3. 匹配字符串操作

strmatch('substr', S)　S 可以是字符串矩阵或由字符串组成的单元阵，如果是单元阵，则必须是单列，函数返回以字符串 substr 开始的行的行号。字符串矩阵的查找速度要比单元阵的查找速度快。

例 63

```
k = strmatch('yes',strvcat('yes','noyes','yesno'))
k = 1                              %只有第 1 行和第 3 行是以“yes”开头的字符串
    3
strmatch('substr',S,'exact')      %返回只精确匹配 substr 的行的行号
```

例 64

```
S={'yes';'noyes';'yesno'}
strmatch('yes',S,'exact')=1       %只有第 1 行精确匹配串“yes”
```

4. 在字符串中查找子串

findstr('str1', 'str2')　此函数在长字符串中查找短的字符串，并返回长字符串中短字符串开始的所有位置。子串和母串在括号中既可在前也可在后，即 str1，str2 中任意一个都可作为子串或母串。

例 65

```
S='What does a good goose look like?'
b=findstr(S,'oo')
b = 14    19    25
```

5. 字符串替换操作

strrep('str1', 'str2', 'str3')　此函数将字符串 str1 中的所有的字符串 str2 用字符串 str3 来代替。其中，str1，str2 和 str3 任何一个可以为字符串组成的单位阵或矩阵，返回的结果与此单位阵或矩阵有相同的大小。如果两个以上为单元阵或矩阵时，则它们的类型和大小必须相同(每行字符数是不同的)。通过例 66 仔细理解此函数的功能。

例 66

```
strrep(S,'oo','ee')
ans =What does a geed geese leek like?

a = 'MatLab'    'welcome'    b = 'MatLab'    'lab'    c='mat'    'come'
        'you'       'me'         'good'    'software'   'you'     'me'
    strrep(a,c,b)                                   strrep(a,'me','you')
```

```
ans =
    'MatLablab'    'wellab'
        'good'        'software'
    strrep('MatLab',b,'!!!')
ans =
   '!!!'       'mat!!!'
       'MatLab'    'MatLab'
    strrep('MatLab','lab',c)
ans =
    'matmat'    'matcome'
        'matyou'    'matme'
```

```
ans =
    'MatLab'    'welcoyou'
             'you'       'you'
    strrep('MatLab',b,c)
ans =
    'mat'       'matcome'
        'MatLab'    'MatLab'
```

6. 得到指定的子串

strtok('string', d) 此函数返回由字符 d 作为分割的字符串 string 的第 1 部分，也就是说，返回字符串 string 中，第 1 个字符 d 之前的所有字符。如果字符串中不含有字符 d，则返回整个字符串；如果 d 字符恰为字符串 string 的第 1 个字符，则函数返回除第 1 个字符之外的所有字符。合法的 d 可以为任意字符或字符串，如果 d 为字符串，则将它的第 1 个字符作为分割符。如果 string 中有前导空格，则前导空格将被忽略。

例 67

```
s='we are good friend ,is that so?'
strtok(s,'at')
ans =we
strtok(s,'w')
ans =e are good friend ,is that so?
strtok(s,'j')
ans =we are good friend ,is that so?
f='  hello world'
strtok(f,' ')
ans =hello
```

strtok('string') 此函数以默认的回车符(ASCII 码为 13)、制表符(ASCII 码为 9)、空格(ASCII 码为 32)作为分割符，前导空格将被忽略。

例 68

```
strtok(f)
ans =hello
```

[token，rem]=strtok(…) 此函数不单返回上面的查找结果 token，还返回剩余的字符串 rem，其中不包括分割符，前导空格被忽略。其中 strtok(…)可以为 strtok('string')或 strtok('string'，d)形式。

例 69

```
[token,rem]=strtok(s,'w')
token =e are good friend ,is that so?
rem =Empty string: 1-by-0
```

```
[token,rem]=strtok(s)
token =we
rem = are good friend ,is that so?
[token,rem]=strtok(f)
token =hello                              %前导空格都已被忽略
rem = world                               %此处的分割空格没被删除
```

7. 判断串中元素是否为字母

isletter(S)　S 可以是字符串或字符串矩阵，此函数返回与 S 同样维数的逻辑矩阵，如果 S 中的元素为字母，则在逻辑矩阵的相应位置上输出 1，否则输出 0。

例 70

```
isletter(f)
ans =0   0   0   0   1   1   1   1   1   0   1 . 1   1   1   1
```

8. 判断串中元素是否为空格

isspace(S)　此函数与 isletter(S)用法相同，在为空格的相应位置上输出 1，否则输出 0。

例 71

```
isspace(f)
ans =1   1   1   1   0   0   0   0   0   1   0   0   0   0   0
```

2.5.4 字符串与数值间的相互转换

MatLab 主要是针对数据或矩阵运算的，因而在对字符串进行操作时必然会经常遇到字符串与数值之间的转换问题。将计算结果按某种格式进行输出，或对图形对象进行标注和说明时就必须将数值转换为字符串。MatLab 提供了将数值转换为字符串和将字符串转换为数值两种功能的函数。

1. 将整数转换为字符串

int2str(A)　其中 A 可以为数或矩阵，当然也包括复数。如果 A 为数，则此函数将 A 转换为字符串；如果 A 为矩阵，则转换为字符串矩阵，每个数之间用空格隔开；如果 A 为复数或复数矩阵，则只将其实部进行转换，即相当于 int2str(real(A))。real(A)为取矩阵 A 的实部，如果 A 中元素不为整数，则先将个数取整，再进行转换。

例 72

```
B=[1.2 3.6 6.7; 3.2 5.5 3.3]
    b=int2str(B)
b =
1  4  7
3  6  3
c=1234.5678
int2str(c) =1235
d = 1.2000 + 3.4000i
```

```
int2str(d)=1
```

2. 将浮点数转换为字符串

num2str 此函数将一个浮点数转换为字符串。这个函数在作图过程中，用相应的计算结果对输出图形进行说明和标注时非常有用，可以用在 M 函数中，根据不同的图形对标注进行相应的变化。

Num2str(A) 此函数将一个浮点数或数组 A 转换为一个字符串或字符串矩阵，如果为复数，则其实部和虚部都不能忽略。

例 73

```
D=[123.11111 3456.777777
-0.00000043567 0.14506]
d=num2str(D)
d =
   '123.11111    3456.7778 '
   '-4.3567e-007      0.14506'
A =
   0.6154 + 1.2000i   0.9218 + 3.6000i   0.1763 + 6.7000i
   0.7919 + 3.2000i   0.7382 + 5.5000i   0.4057 + 3.3000i
a=num2str(A)
a =
'0.61543+1.2i           0.92181+3.6i          0.17627+6.7i  '
'0.79194+3.2i           0.73821+5.5i          0.40571+3.3i  '
size(a) = 2    65                           %为 2×65 的字符串矩阵
```

num2str(A，N)，N 指定了转换的精度，即指定了字符串中每个数字最多包含 N 位数。

例 74

```
a=num2str(A,3)
a =
'0.615+1.2i          0.922+3.6i          0.176+6.7i     '
'0.792+3.2i          0.738+5.5i          0.406+3.3i     '
```

num2str(A，format) 此函数用指定的格式化字符串 format 转换数或矩阵 A。关于格式化输出，格式字符串表示方法与 C 语言相同。

例 75 将 A 按 3 位有效数字的形式输出，每个数字占 10 个字符

```
a=num2str(D,'%10.3g')
a =
'       123 3.46e+003'                 %输出是右对齐的，不足 10 个字符
'-4.36e-007     0.145'                 %用前导空格补齐
```

3. 将字符串转换为浮点数

str2num(S) S 可以为字符串或字符串矩阵，S 必须是合法的数据形式或表达式。如果 S 为表达式，则此函数会给出计算所得的表达式的值，其功能与 feval 函数相同。S 中合法

的字符可以包括：数字 0～9，小数点“.”，正负号“+、-”，表示 10 的乘方的“e”，表示复数虚部的“i”，及各种数学运算符和数学函数计算式，如*，/，sin，log 等。

例 76

```
str2num(a)
    ans =
  1.0e+003 *
    0.1231    3.4568
         0.0000    0.0001
    str2num('sin(2+3)')
ans = -0.9589
str2num('2*4,4/6-7')
ans =-5.5333
h=['2+3   '; 'sin(6)']                         %将字符串矩阵转换为数值矩阵
str2num(h)
ans =                                          %返回计算式的计算结果
    5.0000
   -0.2794
```

从例 76 可以看出此函数的功能非常强大，应当熟练掌握。

2.5.5 二进制、十六进制与十进制间的转换

数据在计算机中是以二进制的形式存在的，而十六进制在实际的表示中比二进制要方便，因而除了十进制以外，二进制数和十六进制数都是比较常用的两种数据表示方法。MatLab 提供了二进制、十进制和十六进制数和字符串之间的转换函数，这些函数在将数据以二进制或十六进制进行格式化输出时是非常有用的。它们之间的转换有以下几种：

1. 把十进制整数转换为十六进制字符串

dec2hex(A) 函数将一个小于 2^{52} 的非负整数转换为其十六进制的字符串形式。

Dec2hex(A，n) 函数将小于 2^{52} 的非负整数 A 转换为 n 位十六进制的字符串形式，如果实际转换成的十六进制数的位数小于 n，则其余位上为 0；如果实际转换成的十六进制数的位数大于 n，则忽略此限制。A 可以为由满足上述条件的整数组成的矩阵，返回结果为字符串矩阵。

例 77

```
a=dec2hex(12334)
a =302E
b=dec2hex(12334,10)
b =000000302E
c=dec2hex(12334,1)
c =302E
A=[1234; 45];
d=dec2hex(A,1)
d = 4D2
02D
```

2. 把十六进制字符串转换为十进制整数

hex2dec(S) 函数将字符串或字符串矩阵表示的十六进制数转换为相应的十进制数。

例 78

```
hex2dec(c)
ans =12334
hex2dec(d)
ans = 1234
        45
```

3. 把十六进制字符串转换为浮点数

hex2num(S) 此函数将字符串表示的十六进制数转换为双精度浮点数。如果输入的字符串少于 16 个字符，函数会用 0 在其后面补足 16 个字符串。S 可以为字符串矩阵。此函数也可以处理 NaN 和 Inf 等数。

例 79

```
hex2num('f')
ans = -3.1050e+231
hex2num('f000000000000000')
ans = -3.1050e+231
hex2num(['03f';'62a'])
ans =
  1.0e+167 *
        0.0000
        1.1794
hex2num('fffffffffffffffffffffffff')
ans = NaN
```

4. 将十进制数转换为二进制字符串

dec2bin(A) 此函数将十进制数或矩阵 A 转换为它的二进制形式的字符串。A 本身或 A 的元素(A 是矩阵时)都必须是小于 2^{52} 的非负整数。

dec2bin(A，n) 此函数将 A 转换成 n 个字符组成的字符串表示的 A 的 n 位二进制数。如果实际转换成的二进制数的位数小于 n，则其余位上为 0，如果实际转换成的二进制数的位数大于 n，则忽略此限制。

例 80

```
A =[8    10; 2     5]
dec2bin(A)
ans ='1000'
'0010'
'1010'
'0101'
dec2bin(A,8)
```

```
ans ='00001000'
'00000010'
'00001010'
'00000101'
```

2.6 函数和特殊函数简明介绍

MatLab 主要进行数学计算，因而各种数学函数在以后的计算中都是必不可少的。常用函数及其功能如下表所示：

2.6.1 三角函数

sin	正弦函数	sinh	双曲函数
asin	反正弦函数	asinh	反双曲函数
cos	余弦函数	cosh	双曲余弦函数
acos	反余弦函数	acosh	反双余弦函数
tan	正切函数	tanh	双曲正切函数
atan	反正切函数	atanh	反双曲正切函数
sec	正割函数	sech	双曲正割函数
asec	反正割函数	asech	反双曲正隔函数
cot	余切函数	costh	双曲余切函数
acot	反余切函数	acoth	反双曲余切函数

2.6.2 其他常用计算函数

fix	超零方向取整	roud	四舍五入到最近的整数
floor	超无穷方向取整	rem	求两整数相除的余数
ceil	超正无穷方向取整	exp	求指数函数
log	自然对数	log10	求以 10 为底的函数
sort	求数值的平方根	abs	求绝对值
conj	求复数的共轭	imag	求复数的虚部
real	求复数的实部		

2.6.3 常用的矩阵函数

sqrtm	求矩阵的平方根	expm	求矩阵的指数值
funm	求按矩阵计算的函数值	logm	求矩阵的对数值

以上函数的基本功能在此不再赘述。

2.7　M 文件与 M 函数

MatLab 输入命令的方式大体有两种，一种是在工作空间中直接输入简单的命令，这种方式适应于命令行大都比较简单，输入比较方便，且处理的问题比较特殊、没有一定的重复性和普遍应用性、且差错处理比较简单方便的场合。上面所列举的例子中绝大多数都是采用这种输入方式，这种方式主要体现了 MatLab 作为一种“数学演算和图视工具”的特点，在以上的学习中都已经体会到这种工作方式的优越性和方便之处，这正是其他数学计算工具和编程语言所无法比拟的。然而单有这方面的功能还不足以体现 MatLab 的功能的完整性和易于使用的特点。在进行大量重复性的计算和输入时，单靠直接输入是非常繁琐的。而 MatLab 提供了另一种功能强大的工作方式，这就是 M 文件的编程工作方式。M 文件的语法类似于一般高级语言，是一种程序化的编程语言，但 M 文件又有其自身的特点。它只是一个简单的 ASCII 码文本文件，语法比一般的高级语言都要简单，程序容易调试，交互性强。MatLab 是解释性的编程语言，即逐句解释运行程序。Matalab 在初次运行 M 文件时将 M 文件编程代码并装入内存中，此过程会大大降低程序的运行速度，但再次运行该程序时便会直接从内存中取出代码运行，会大大加快程序的运行。MatLab 的 M 程序是注重于数学计算的一门编程语言，直接采用复数矩阵作为基本的运算单位，因而不论从形式上还是从语法上来说都是比较简单易学的，而且比较容易维护。且由于 MatLab 是用 C 语言编写而成的，与 C 语言有着千丝万缕的联系，熟悉 C 语言则更易学习 M 文件。

M 文件可以像一般的文本文件那样在任何文本编辑器中进行编辑、存储、修改和读取，利用 M 文件，可以自编函数和命令，对已经存在的命令和函数进行扩充和修改，因而对 MatLab 进行二次开发是非常方便的。M 文件有两种形式，一是命令文件，或称作脚本文件；另一种是函数文件。两种文件的扩展名都是.m。

2.7.1　命令文件

如果要输入较多的命令，而且要经常对这些命令进行重复输入，利用命令文件就显得比较简单和方便。可以将要重复输入的所有命令按顺序放到一个扩展名为“.m”的文本文件中，每次运行时只要输入 M 文件的文件名即可。注意：此 M 文件要放在 MatLab 的搜索路径下，且文件名最好不要与 MatLab 的内置函数和工具箱中的函数重名，以免产生混淆和发生执行了错误命令等错误。事实上，命令文件与 DOS 下的批处理命令，或其他语言的一些脚本文件，如宏命令等有相似之处，可以借助这些比较熟悉的语言来加深对命令文件的理解。MatLab 对命令文件的执行就是等价于从命令行窗口中顺序执行文件中的所有指令。

命令文件中的语句可以访问 MatLab 的工作空间中的所有变量和数据，在命令文件运行过程中产生的所有变量都是等价于直接从 MatLab 工作空间中建立，因而任何其他的命令文件和函数都可以访问这些变量。这些全局变量产生后就一直保存在内存中，可以用 clear 命令来清除工作空间中的变量，包括全局变量。

例 81 编一个命令文件，产生 10 个变量，变量值为随机数。文件名为 example1.m。

(1) 在命令行中输入 edit，依次输入下列各行：

```
%这是一个 M 文件的例子，用来建立 10 个全局变量
%变量名是 a1~a10
for i=1:10
  eval(['a' int2str(i) '=rand;']);
end
%display the result
disp('the values of the ten variables are:')
for i=1:10
  eval(['a' int2str(i)]);
  end
```

(2) 在命令行中输入"example1"，运行结果为：

```
a1 = 0.3529
a2 = 0.8132
a3 = 0.0099
a4 = 0.1389
a5 = 0.2028
a6 = 0.1987
a7 = 0.6038
a8 = 0.2722
a9 = 0.1988
a10 = 0.0153
help example1          %显示建立的关于 example1.m 的帮助文件
```

这是一个 M 文件的例子，用来建立 10 个全局变量，这 10 个全局变量如下：

```
变量名是 a1~a10
who                   %查询当前工作空间中的变量
Your variables are:
a1      a2      a4      a6      a8      i
a10     a3      a5      a7      a9
```

例 82 中，文件名为 leapyear.m，是用来找出从 1～1000 年共有哪些年是闰年。

例 82 程序的清单如下：

```
  flag=0;
for year=1:1000
      if rem(year,4)==0                    %从此开始是判断闰年
    if rem(year,100)==0
     if rem(year,400)==0
       leap=1;
     else
       leap=0;
     end
   else
```

```
            leap=1;
          end
        else
          leap=0;
        end                          %判断闰年结束
        if leap
          disp([int2str(year) 'is a leap year']);     %显示哪年是闰年
          flag=flag+1;
        end
    end
    if flag==0                       %如果没有闰年，则显示下列信息
      disp('there is no leap year from 1 to 1000');
    else                             %如果找到闰年，则显示闰年的总数
      disp(['there are totaly ' int2str(flag) 'leap years from 1 to 1000']);
    end
```

执行结果为：

```
4is a leap year
8is a leap year
12is a leap year
16is a leap year
⋮   ⋮ ⋮  ⋮
996is a leap year
there are totaly 242leap years from 1 to 1000
```

说明：在命令文件中，由符号"%"来说明从此符号开始到本行末都是注释行，MatLab将予以忽略而不执行。由命令文件创建的变量都是当前工作空间中的全局变量，可以通过who和whos等命令进行查看。一定要保证自己创建的命令文件在MatLab的搜索路径下，可以通过1.4节介绍的path命令或界面操作将自创的任意一个目录加入MatLab的搜索路径下。

2.7.2 函数文件

一般函数文件的第1行都是以function开始，说明此文件定义的是一个函数。函数文件实际上定义的是一个MatLab的子函数，其作用与其他高级语言的子函数基本相同，都是为了方便地实现功能而定义的。MatLab本身提供了许多工具箱，如信号处理、图像处理、样条分析、神经网络、金融财政等，扩展了MatLab的功能，提供了许多有用的函数，其中大都是由M文件的函数形式定义的。函数文件是扩展MatLab本身的功能并对其进行二次开发的强有力的工具。

函数文件与命令文件的主要区别在于：函数文件一般都要带参数，都要有返回结果，也有一些函数文件不带参数和返回结果，而且函数文件要定义函数名；而命令文件没有参数和返回结果，也不在程序的开头定义函数名，通过生成和访问全局变量可以与外界和其他函数交换数据。命令文件的变量在文件执行结束后仍然会保存在内存中不丢失；而函数文件的变量仅在函数运行期间有效，当函数运行完毕，它所定义的所有变量都会被清除，

即用 who 命令不会看到这些变量，而在函数运行期间运行 who 命令只会看到函数内部定义的局部变量和传递的参数及返回变量。函数中定义的变量可以与当前工作空间中的变量重名，而不会影响工作空间中原来变量的存储。为了更好地理解这些概念，请认真阅读下面的例子。

例 83

(1) 在当前工作空间中建立以下 3 个变量：

```
a1 =0.3529
a2 =0.8132
a3 = 0.0099
```

(2) 定义函数文件名为 example2.m，其内容如下：

```
function y=example2(n)
%函数用法 y=example2(n)，其中 n 为任意数和矩阵
%此函数用来检验函数中的局部变量同工作空间中变量之间的关系
a1=n;          %对 3 个局部变量进行赋值
a2=n+1;
a3=n+2;
whos           %详细查看当前的所有变量
a1             %显示 3 个变量的值
a2
a3
y=a1+a2+a3     %给返回的变量赋值
```

(3) 执行结果如下：

```
example2(2)
  Name      Size        Bytes  Class
  a1        1x1             8  double array
  a2        1x1             8  double array
  a3        1x1             8  double array
  n         1x1             8  double array
%因为当前变量 y 还未被赋值，因而没有显示出变量 y
Grand total is 4 elements using 32 bytes
a1 = 2
a2 =3
a3 =4
y = 9
ans = 9       %此处为函数 example2(2)的返回结果
```

(4) 当程序执行完毕后，再查看一下变量 a1，a2，a3 的值，可以发现，它们的值并没有发生变化，仍然为：

```
a1 =0.3529
a2 =0.8132
a3 = 0.0099
```

(5) 如果对上面的程序进行修改，将函数文件变成命令文件，名为 example3.m。内容如下：

```
%此函数用来检验函数中的局部变量同工作空间中变量之间的关系
a1=2;       %对 3 个局部变量进行赋值
a2=3;
a3=4;
whos        %详细查看当前的所有变量
```

(6) 执行结果如下：

```
example3
Name      Size        Bytes  Class
a1        1x1            8  double array
a2        1x1            8  double array
a3        1x1            8  double array
ans       1x1            8  double array
c         1x1            8  double array (global)
flag      1x1            8  double array
i         1x1            8  double array
j         1x2          200  cell array
leap      1x1            8  double array
p         1x1415      2830  char array
s         2x3           12  char array
year      1x1            8  double array
```

上面的显示结果中有许多变量并不是在命令函数中定义的。

(7) 执行完毕后分别查看下面的变量的值，可以看出已经变为：

```
a1 =2
a2 =3
a3 = 4
```

例 84　编一函数搜索含有字符串 S 的路径的全称

程序名为 findpath.m，程序清单为：

```
function findpath(S)
%此函数用来搜索包含字符串 S 的路径，并将所有符合条件的路径的全部显示出
%来，命令的格式是 findpath('string')，例：
%findpath('map')     则程序显示：
  %the directory you want is: d:\MatLab\toolbox\map\map
%the directory you want is: d:\MatLab\toolbox\map\mapdisp
%the directory you want is: d:\MatLab\toolbox\map\mapproj
if ~isstr(S)             %判断 S 是否为字符串，如果不是则显示出错信息
      error('the parameter should be a string');
   return;
end
currentpath=path;
while 1
 [token,currentpath]=strtok(currentpath,';');
if all(size(currentpath))
   currentpath(1)=[];
end
if ~isempty(findstr(token,S))
```

```
        disp(['the directory you want is: ' token]); %显示查询结果
    end
    if ~all(size(currentpath))          %判断 currentpath 是否为空,
                                        %如果为空就停止循环
        return;
        end
    end
```

如要查找包含字符串' graph'的所有路径，函数执行及返回结果为:

```
findpath('graph')
the directory you want is: d:\MatLab\toolbox\MatLab\graph2d
the directory you want is: d:\MatLab\toolbox\MatLab\graph3d
the directory you want is: d:\MatLab\toolbox\MatLab\specgraph
the directory you want is: d:\MatLab\toolbox\MatLab\graphics
```

例 84 中的程序只适应于单个字符串，稍加修改，就可以适用于字符串矩阵和由字符串组成的单元阵。望自行完成。

说明：定义函数的一般步骤是：首先在函数文件的第 1 行第 1 个函数名中定义输入参数和输出参数，分别为 n 和 y，参数类型视具体情况而定。然后输入程序的具体语句，并对输出参数进行赋值。“%”后面的语句为注释行，对程序起解释说明的作用，而且还对 help 命令及 lookfor 命令起在线查询作用。

例 85 如要查询函数 example2

```
help example2
```

y=example2(n) 此函数中 n 为任意数和矩阵。用来检验函数中的局部变量同工作空间中变量之间的关系。

用 lookfor 命令进行查询:

```
lookfor example2
```

example2.m 函数用法：y=example2(n)，其中 n 为任意数和矩阵

可以看出，lookfor 命令只对注释行的第 1 行进行查询。

有一些函数会出现多个返回结果，其基本的思想和定义过程与单返回结果的函数基本相同，只是在定义函数时要同时定义多个返回变量。也可以在调用函数时只要求返回前面的结果。举例说明如下。

例 86 建立下面的函数，名为 example4.m

```
function [a,b,c]=example4(n)
%函数调用为 example4(n)
%此函数是有多个返回结果的函数
a=n.^n;
b=n.^n-n;
c=sqrt(abs(n));
```

执行结果如下：

```
a=[2 4; -3 5]
example4(a)=
 1.0e+003 *
  0.0040    0.2560
  0.0000    3.1250
[a1,a2]=example4(a)
a1 =                                      a2 =
  1.0e+003 *                              1.0e+003 *
  0.0040    0.2560                        0.0020    0.2520
  0.0000    3.1250                        0.0030    3.1200
[a1,a2,a3]=example4(a)
a1 =                                      a3 =
 1.0e+003 *
  0.0040    0.2560                    1.4142    2.0000
  0.0000    3.1250                    1.7321    2.2361
a2 =
 1.0e+003 *
  0.0020    0.2520
  0.0030    3.1200
```

2.7.3　函数的调用

函数的调用一般分为嵌套调用和递归调用。被调用的函数必须为已经存在的函数，包括 MatLab 的内嵌的库函数。函数的调用可以为单层的，也可以为多层的，在同一个函数中可以多次递归或嵌套调用相同或不同的函数。

1.　函数的嵌套调用

一个 M 函数可以调用任意其他的函数，这叫函数的嵌套调用；被调用的函数又可以调用其他的函数，这叫函数的多层嵌套调用。只要编写 M 函数，就要用到函数的嵌套调用，上面所编写的任何 M 文件都要调用其他的 M 函数或 MatLab 的内部函数。几乎所有的计算机语言都允许对函数进行嵌套调用，否则，编写函数将非常复杂。

2.　函数的递归调用

在调用一个函数的过程中又出现直接或间接地调用该函数本身的现象，叫函数的递归调用。并不是所有的计算机语言都允许函数的递归调用，C 语言是允许的，但是 FORTRAN 语言就不允许。要充分理解递归调用的概念，它在解决许多问题时是非常有效的。如果函数 A 在执行过程中直接调用函数 A 本身，按照定义，这当然是函数的递归调用；如果函数 A 在执行过程中并没有直接调用其本身，而是调用了函数 B，而函数 B 在执行过程中却调用了函数 A，这也是函数的递归调用。在递归调用的函数中一般要有跳出递归调用的语句，否则函数会无穷循环下去。下面举例 87 说明函数的递归调用。

例 87

```
function y=fac(n)
if n<0
   error('n is smaller than 0,please check it !');
   return;
end
if n==0|n==1
   y=1;
else
   y=n*fac(n-1);                          %对函数本身进行递归调用
end
```

例 87 的程序中，如果 n 是大于 1 的整数，函数会持续递归调用函数本身，每调用一次，参数 n 的值就减去 1，直到 n 的值为 1，到此函数的递归调用过程结束。函数的执行结果为：

```
fac(12)
ans =
   479001600
```

2.8 程序结构与控制

程序一般分为顺序结构、循环结构、分支结构 3 种，这是与大多数其他高级语言一致的，理论上说来，计算机语言只要有以上 3 种结构就可以构造功能强大的程序。由于 MatLab 是由 C 语言编成，它的控制语句也同 C 语言有相似之处，但是又有不同之处。MatLab 的控制语句没有 C 语言那么复杂、灵活和多变，因而语法等都比较简单，容易掌握。但 MatLab 自身的强大功能弥补了这个不足，使这些语句仍然适用。下面将分别对这 3 种结构进行介绍。

2.8.1 顺序结构

顺序结构就是依次顺序执行程序的各条语句。语句在程序文件中的物理位置就反映了程序的执行顺序。一个典型的顺序结构，就是不含有其他子结构和控制语句的批处理文件或是 MatLab 中的命令文件。虽然大多数程序都包含许多子结构，但是从整体上看，它们大都遵循顺序结构。例 88 就是一个典型的顺序结构的程序：

例 88 命令文件 example5.m

```
disp('the begin of the program')
disp('the first line')
disp('the second line')
disp('the third line')
disp('the end of the program')
```

执行结果为：

```
the begin of the program
```

```
the first line
the second line
the third line
the end of the program
```

可见，命令依次显示各条语句，即依次执行各条命令。

2.8.2 循环结构

循环是计算机解决问题的主要手段，许许多多实际问题大都包含有规律性的重复计算和对某些语句的重复执行。循环结构中，被重复执行的那一组语句就是循环体，每个循环语句都要有循环条件，以判断循环是否要继续进行下去。MatLab 的循环语句主要有 for-end 和 while-end 语句。

1. for-end 循环控制

for 循环将循环体中的语句重复执行给定的次数，循环的次数一般情况下是已知的，除非用其他的语句将循环提前结束。for 循环的语法为：

```
for i=表达式
    可执行语句 1
        ……
    可执行语句 n
end
```

表达式是一个向量，可以为 m:s:n，m，s 和 n 都可以为整数、小数，还可以为负数。不论它们取何值，都必须满足构成向量的条件，如 1:-0.5:3 就不能构成向量，所以不符号条件；而 3.5:-0.3:-9 就符合构成向量的条件。此向量中的元素被逐一赋值给 i，对每个 i 的不同取值都要执行一次循环体内的语句。表达式也可以为 m:n，此时，默认的步长为 1，因而 m 和 n 必须满足 m<=n 的条件。因为 i 取向量 m:s:n 中的所有取值，可以通过直接将一个向量赋值给 i，使 i 穷尽此向量中的取值。i 也可以为字符串、字符串矩阵、或字符串组成的单元阵，i 将依次穷尽其中的所有元素，但作为取值的单元是有所不同的，如字符串是以每个字符作为取值单元的，单元阵是以单元阵的元素为取值单元的，数值矩阵是以列向量作为取值单元的，等等。

例 89

(1)
```
for i=9.8:-3:-9
        i
    end
```

显示结果为：

```
i = 9.8000, i = 6.8000, i =3.8000, i = 0.8000, i =-2.2000, i = -5.2000,
i = -8.2000,
```

(2)
```
a =  1   3   5
     6   7   9
```

```
for i=a
i
end
i = 1          i=  3                i=  5
    6               7                    9
```

(3) s='abcdefghilh'

```
for i=s
i
end
i =a, i =b, i =c, i =d, i =e, i =f, i =g, i =h, i =i, i =l, i =h
```

(4) c={'aaa' 'bbbb' 'ccc'}

```
for i=c
i
end
i =  'aaa',          i =  'bbbb',          i =  'ccc'
```

(5) S=['aaa';'bbb';'ccc'];

```
for i=S
i
end
i =                          i =
i=
a                            a
a
b                            b
b
c                            c
c
```

例 90 利用 for 循环求 1～100 的整数之和

```
sum=0;
for i=1:100
sum=sum+i;
end
sum =  5050
```

for 循环的循环体中，可以多次嵌套 for 和其他的结构体，这在一些场合是非常有用的，大大扩展了 for 循环的用途。

例 91 利用 for 循环求 1!+2!+3!+…+20!的值

程序如下：

```
sum=0;
for i=1:20
  prd=1;
  for k=1:i
```

```
      prd=prd*k;
    end
    sum=sum+prd;
  end
  sum
```

执行结果为：

```
sum =2.5613e+018
```

例 92　利用 for 循环找出从 100～200 之间的所有素数

```
for m=101:2:200
   k=fix(sqrt(m));
   for i=2:k+1
      if rem(m,i)==0
         break;
      end
   end
   if i>=k+1
      disp(int2str(m))
   end
end
```

运行结果为：

```
 101 103 107 109 113 127 131 137 139 149 151 157 163 167 173 179 181
191  193 197 199
```

2. while-end 循环

while 循环将循环体中的语句循环执行不定次数。其基本语法为：

```
while  表达式
       循环体语句
end
```

表达式一般是由逻辑运算和关系运算以及一般的运算组成的表达式，以判断循环要继续进行还是要停止循环。只要表达式的值非零，即为逻辑“真”，程序就继续循环，只要表达式的值为 0 就停止循环。

例 93　用 while 循环求 1～100 间的整数的和

```
sum=0;
i=1;
while i<=100
sum=sum+i;
i=i+1;
end
sum
sum = 5050
```

while 循环和 for 循环是可以相互转化的。将 for 循环转化为 while 循环，可以这样改写：

```
i=m;
while i<=n
{for 循环体}
i=i+s;
end
```

将 while 循环转化为 for 循环，可以这样改写：

```
for i=1:inf
 if ~(while 循环中的表达式)
    break;
 end
 {while 循环体}
end
```

例 94 用 while 循环改写例 92

```
m=101;
while m<=200
   k=fix(sqrt(m));
   i=2;
   while i<=k+1
      if rem(m,i)==0
         break;
      end
      i=i+1;
   end
   if i>=k+1
      disp(int2str(m))
   end
   m=m+2;
end
```

2.8.3 分支结构

在计算中通常要根据一定的条件来执行不同的语句，当某些条件满足时只执行其中的某一条或某几条命令，在这种情况下就要用到分支语句。MatLab 提供了 2 种分支语句，一种是 if-else-end，另一种是 switch-case-end 语句，两者各有特点，下面分别介绍。

1. if-else-end 分支结构

此分支结构一般有 3 种形式：

```
(1) if  表达式
        执行语句
 end
```

如果表达式的值非 0，则执行下面的执行语句，否则跳过此语句转而执行 end 后面的语句。如：

```
if  i>j
        disp([int2str(i)       '>'       int2str(j)    ]),disp(['i='
int2str(i)]),disp(['j='  int2str(j)])
    end
```

说明：多条语句可以放到同一行里，但每行语句之间要用“，”或“；”隔开。

```
(2) If  表达式
语句 1
else
语句 2
end
```

如果表达式值为非 0 值，就执行语句 1，否则就执行语句 2。

```
(3) If  表达式 1
语句 1
elseif  表达式 2
语句 2
elseif  表达式 3
语句 3
┆   ┆
else
语句 n
end
```

此过程较复杂，程序先判断表达式 1，如果表达式 1 成立，则执行语句 1，执行完语句 1 后便终止执行 if 语句，即使后面的表达也有可能成立，也会被忽略掉。如果表达式 1 不成立，就对表达式 2 进行判断，如果此表达式成立，就执行语句 2，且忽略后面的语句；如果不成立，则继续这个过程进行判断.如果所有的表达式都不成立，就执行 else 后面的执行语句(如果 else 语句存在的话)。

说明：else 语句也可以不存在。

例 95　对于下面的语句：

```
  if i<0
disp('i<0')
  elseif i<1
disp('i<1')
  elseif i<2
disp('i<2')
  else
disp('i>=2')
  end
```

如果 i=-1，则执行结果会显示：i<0，虽然此时 i 既小于 0 又小于 1"又小于 2，所有表达式也都是成立的，但程序只执行第 1 条成立的表达式后面的语句，其余语句将被忽略。

在上面所有条件表达式中，通常都是由关系操作符(如>，<，<=，>=，==，~=等)、逻辑操作符(如&，|，~等)和逻辑函数(如 isequal，isempty，isstr 等)等组成的表达式，表达式中可以只含有标量间的运算，还可以是矩阵间的运算。

例 96

```
A=[1    2; 0    1]
    B=[2    2; 3    3]
    if A<B
    disp('A<B')
    elseif A<B+1
    disp('A<B+1')
    end
    A<B+1                    %显示的结果
```

可见，A<B 不成立，而 A<B+1 成立，在 if 表达式中，A<B+1 成立就相当于 all(all(A<B+1))=1。

例 97 编一函数计算函数值：

$$\begin{cases} x & (x<1) \\ 2x-1 & (1<x<10) \\ 3x-11 & (x>=10) \end{cases}$$

```
  function  y=yx(x)
if x<1
   y=x;
elseif x>=1&x<10
   y=2*x-1;
else
   y=3*x-11;
end
```

2. switch-case-end 分支结构

switch 语句是多分支选择语句，虽然在某些场合 switch 的功能可以由 if 语句的多层嵌套来完成，但是会使程序变得复杂和难于修改维护，而利用 switch 语句构造多分支选择时显得更加简单明了、容易理解。

switch 的基本用法为：

```
switch 表达式
case 常量表达式 1
     语句块 1
case 常量表达式 2
     语句块 2
case {常量表达式 n, 常量表达式 n+1, 常量表达式 n+2…}
     语句块 n
```

```
otherwise
    语句块 n+1
end
```

- switch 后面的表达式可以为任何类型，如字符串、矩阵等。
- 当表达式的值与 case 后面的某个常量表达式的值相等时，就执行这个 case 后面的语句块，如果所有的常量表达式的值都与此表达式的值不匹配时，就执行 otherwise 后面的语句块。
- 与 if 语句不同的是，各个 case 和 otherwise 语句出现的先后顺序并不会影响程序的执行结果，将 otherwise 语句放到 case 语句的前面也是合法的。
- 每个 case 后面的常量表达式可以有多个，而且可以是不同的类型，每个 case 中常量表达式的值是可以重复的，这在语法上没有错误，只是在执行时后面那些符合条件的 case 语句将被忽略，不会起作用。
- 每次只执行一个语句块，执行完一个语句块后就退出 switch 语句。

例 98 执行下面的语句(文件名为 caseme.m)：

```
switch var
case {'aaa','bcd',1}, disp('the first case line')
case 3, disp('the second case line')
case {2,3,4,'welcome'}, disp('the third case line')
case {5,6,7,3,'MatLab'}, disp('the forth case line')
otherwise, disp('otherwise line,MatLab is a good software')
end
```

如果 var =3，执行后只会显示 the second case line 而不会显示 the third case line。

如果 var='kkk'，则显示 otherwise line，MatLab is a good software。

在例 98 中，如果 case 后面的常量表达式是一个单元阵，则会将表达式的值与单元阵中的所有元素进行比较，如果表达式的值与其中任意一个元素相匹配，就执行此 case 语句后面的语句块，并忽略其他的可执行语句块。

例 99 我国新税法规定：个体工商户的生产、经营所得和对企事业单位的承包经营、承租经营所得应缴纳的个人所得税为

	全年收入中应纳税所得税部分	税率(%)
1	不超过 5000 元的	5
2	超过 5000 元至 10000 元的部分	10
3	超过 10000 元至 30000 元的部分	20
4	超过 30000 元至 50000 元的部分	30
5	超过 50000 元的部分	35

请编程序加以计算。

```
function  y=shui(x)
n=fix(x/1000);
switch n
case {0,1,2,3,4}
  y=x*0.05;
```

```
case {5,6,7,8,9}
   y=x*0.1;
case {10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29}
   y=x*0.2;
case {30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49}
   y=x*0.3;
otherwise
   y=x*0.35;
end
```

2.8.4 程序流控制及其他

在 MatLab 的程序设计中，有时候需要提前终止循环、跳出子程序、显示出错或警告信息、显示批处理文件的执行过程等，就要用到程序特殊流程控制命令。MatLab 中的流程控制及错误显示语句有以下几条：

1. 中断当前流程控制结构

break 语句通常用于循环控制中，如 for，while 等循环，通过 if 语句判断是否满足一定的条件，如果条件满足就调用 break 语句，在循环未自然终止之前跳出当前循环。在多层循环嵌套中，break 只是终止包含 break 指令的最内层的循环。

在例 92 中，有两层循环嵌套，最内层的循环用一个 if 语句来判断数 m 是否能被数 i 整除，如果 m 能被 i 整除,就会终止当前最内层的循环,但是外层关于 m 的循环是不会被此 break 语句来终止的。也就是说，break 语句只会终止包含它的最内层的循环。

2. 函数返回调用

return 使当前正在运行的函数正常结束并返回调用它的函数继续运行，或返回到调用它的环境，如命令窗口。这个函数通常用在函数里面，对输入的参数进行判断，如果参数不符合要求，就调用 return 语句终止当前程序的运行，并返回调用它的函数或环境。

例 100 建立以下两个函数：

```
function goodnum(n)
disp('the begin of goodnum function');
if n>=0
disp('n is a good number');
else
return;
end
disp('the end of goodnum function');
function return2(n)
disp('the begin of return2 function');
if n>=2
goodnum(n);
return;
else
```

```
goodnum(n);
end
disp('the end of return2 function');
```

执行结果如下，注意return在这2个函数中起的作用及对程序执行过程的影响：

```
return2(2)
the begin of return2 function
the begin of goodnum function
n is a good number
the end of goodnum function
return2(-1)
the begin of return2 function
the begin of goodnum function
the end of return2 function
```

3. 结构块的界定

end语句主要用来结束for，while，switch，if等结构。end要与这些关键词之间相互匹配，否则会出错或不按程序的本意执行。在较长的程序中出现这样的错误比较难查找，但是利用MatLab自带的Edit编辑器会自动缩进语句，并以不同的颜色显示不同作用的字符，便于查找差错。

4. 显示文本和矩阵

disp(X) 此函数显示字符串、字符串矩阵、数值矩阵、单元阵等，在显示一个变量的内容时，其结果与在命令行中直接输入变量名所得结果基本相同，但并不显示变量名。这个函数也在M文件中用来显示各种运算结果。

例101

```
A =[0.9501    0.6068    0.8913;0.2311    0.4860    0.7621]
    disp(A)                                   %显示结果中并没含有A=的字样
   0.9501    0.6068    0.8913
0.2311    0.4860    0.7621
C={1,2,'hello'; 'MatLab',3,6}
disp(C)
         [1]    [2]    'hello'
   'MatLab'    [3]    [6]
```

5. 显示M文件的执行过程

echo命令通常用来控制M文件在执行过程中显示与否。在正常的执行过程中，M文件是不会显示在命令窗口中的，但在特殊的场合，如需要对M文件进行调试和演示时，则需要M文件执行的每条命令都显示出来，用此命令就可以实现这样的操作。

对于命令文件和函数文件，echo 命令作用的方式有所不同。对于命令文件，其用法比较简单，每个命令都是对所有的命令文件起作用的：

```
echo on       打开所有命令文件的显示方式
echo off      关闭所有命令文件的显示方式
echo          在以上两者间进行切换
```

例 102　打开显示方式，执行上面的 caseme.m 的命令文件

```
var=3
echo on
caseme
switch var                              %从此行开始是显示结果
the second case line
case {2,3,4,'welcome'}, disp('the third case line')
```

以上命令只会影响命令文件的显示方式，而对于函数文件则不起作用。对于函数文件，命令比较复杂，如下：

```
echo filename on         打开名为 filename 的函数文件的显示方式
echo filename off        关闭名为 filename 的函数文件的显示方式
echo filename            切换名为 filename 的函数文件的显示方式
echo on all              打开所有函数文件的显示方式
echo off all             关闭所有函数文件的显示方式
```

对于所有函数文件的操作指的是当前在内存中的所有函数文件，而对于那些没有装入内存中的函数文件仍然不起作用。可以用 inmem 命令查看当前内存中有那些函数。

例 103　对于函数 yx(x)

```
    function  y=yx(x)
    if x<1
        y=x;
    elseif x>=1&x<10
        y=2*x-1;
    else
        y=3*x-11;
    end
inmem                         %查看当前内存中的函数
ans =                         %可见，没有函数 yx
    'goodnum'
    'return2'
'MatLabrc'
echo on all                   %此命令不会影响函数 yx
yx(4)                         %求 yx(4)的值，此时函数 yx 被装入内存
ans =  7
echo on all                   %此命令将影响函数 yx
yx(9)                         %求 yx(9)的值
if x<1                        %请对照上面函数的定义查看函数的执行过程
elseif x>=1&x<10
```

```
    y=2*x-1;
  ans =17
```

在显示方式下，函数文件是逐行解释执行的，而不是编译执行的，这将大大降低程序的执行速度和效率，除非有必要，如在调试时，否则最好不要打开函数文件的显示方式。

6. 显示错误信息

error('错误信息')　此命令显示错误信息并终止当前函数的运行，并将控制返回到键盘。与 break 及 return 的区别是：break 是终止所在的最内层的循环；return 是终止其所在的函数的运行，并将控制返回上一级函数或系统；而 error 是终止当前正在运行的程序并将控制返回到键盘，不论它在哪一层函数中被调用，都会终止整个程序。如果“错误信息”是个空串，则 error 命令不起作用。

2.9　习　　题

(1) 分别用矩阵运算及数组运算，计算下面一组矩阵的加、减、乘、除运算结果，并分析利用矩阵运算与利用数组运算计算所得的结果有什么不同。

```
[2  3  4                [9  0  1
 4  2  8                 0  1  3
 0  4   1]               9  3  0]
```

(2) 用 format 的不同格式显示变量 pi，并分析各个格式之间有什么相同和不同之处。

(3) 利用公式 $\pi/4=1-1/3+1/5-1/7+\cdots+1/20$ 来求 π 的近似值。

(4) 编函数计算 $1!+2!+3!+\cdots+30!$

(5) 编函数统计一个字符串中的单词的个数。

(6) 编写一个命令文件，统计工作空间中已经存在的一个字符串中的字母、数字、空格以及其他字符的个数，并将统计结果保存在一个矩阵变量中。

(7) 用递归调用法求 n 阶勒让德多项式的值，其递归调用的公式为：

```
p(n,x)=((2*n-1)*x*p(n-1,x)-(n-1)*p(n-2))/n
p(0,x)=1;p(1,x)=x;
```

(8) 有两个矩阵 A 和 B 如下：

```
A=[1  4  6  8                B=[-3  -7  6  -5
   2  3  7  0                    -2  1  3  -1
   -2 -1 5  7                    0   0  1  3
   1  1  -1 0]                   2  6  -1  0]
```

将 A 中所有等于-1 的元素改为-2，将 B 中所有小于 0 的元素改为 1，然后将 B 中等于 0 的元素的值改为 A 的相应位置元素的值。

(9) 对上面的两个矩阵求以下关系运算，理解关系运算的含义：

① A>B
② A==0
③ A>1&A<7

(10) s1='this is an example of MatLab   '，将上面的字符串中的后面的空格删除，将大写字母转化为小写字母，并利用关系函数统计其中的空格的个数，然后查找子串“matlab”并将其替换为“MATLAB”。

第 3 章　MatLab 符号运算及数值运算操作

在数学运算中除了数值计算以外，在数学、物理、应用工程和科学方面的抽象运算，即计算式中带有 x，y，g，α，β 等符号变量、表达式的运算，也占了相当大的比例。在一般的程序设计软件如 C，C++，FORTURN 等语言平台上实现数值计算还可以，但要实现抽象运算即符号运算，则不是一件容易的事。而软件 MAPLE V 主要针对符号运算，它具有强大的的符号运算能力，MatLab 在扩展符号运算功能包时，就是靠 MAPLE 实现的。1993 年 MatLab 的开发和销售商 MathWorks 公司购得了 MAPLE 使用权。随后，MathWorks 公司以 MAPLE 的"内核"为符号计算的"引擎"，依靠 MAPLE 已有的库(Library)，开发了在 MatLab 环境下实现符号计算的工具包 Symbolic Math Toolbox。

MAPLE V 软件提供的数学函数达 2000 多种，涉及了数学中许多既有代表性又有实用性的领域：基本代数学、欧几里德几何学、数论、有理函数、微积分、线性代数及矩阵论、微分方程、图形学、离散数学、群论……

在 1994 年 1 月推出的 Symbolic Math Toolbox 第 1 版是以 MAPLE V2 为基础开发的，大概需 10M 的空间。而目前最新的 MatLab 5.2 版本以 MAPLE V4 为其开发基础，大概需 12.2M 的空间。

在 MatLab 的符号运算工具箱 Symbolic Math Toolbox 中，有两个子工具箱：基本符号运算工具箱和扩展符号运算工具箱。其中，基本符号运算工具箱是 MatLab 语言的一种自然延伸。在这个工具箱内，集中了 100 余个 MatLab 函数，这些函数为调用 MatLab 符号运算的“引擎”，即 MAPLE V4 的"内核"提供了相应的命令。而且可在这个工具箱内调用 MAPLE V4 的线性代数工具包。符号运算工具箱的另一个工具箱——扩展符号运算工具箱的功能是：在这个工具箱内，可以调用所有非图形类的属于 MAPLE V4 的工具包，运用 MAPLE V4 的编程特征完成自设的运算。在这两个子工具箱的辅助下，可以编写自己的 M 文件和函数

Symbolic Math Toolbox 有 3 个通道与 MAPLE 交换信息：

- 在用 MatLab 语言写的诸多函数中，通过若干个专用函数进行符号运算。这些专用函数按内容可分为：
    - 符号表达式与矩阵的操作
    - 线性代数
    - 微积分
    - 方程的求解
    - 化简、展开与带入

  - ◆ 特殊数学函数
- 通过 maple.m，mpa.m 两个专门设计的 M 文件进行符号运算。这种符号运算的运算方式，要求掌握一些 MAPLE 的基本语句。
- 通过 MatLab 中的函数计算器 Function Caculator 来进行简单的符号运算。这是最方便且最直观的符号运算方法。

符号计算部分是本书最重要的部分。需要注意的是，在符号运算的整个计算过程中，所有的运算均是以符号进行的，即使以数字形式出现的量也是字符量。

3.1 创建符号变量

在 MatLab 的数据类型中，符号型与字符型是两种重要而又容易混淆的数据类型。符号运算工具箱中的一些命令，它们的参数既可以是符号型，又可以是字符型；而还有很多命令，它们的参数则必须是非符号型。鉴于符号型数据是符号运算的主要数据类型，因此在说明完两种数据类型变量的创建方法和不同之处后，将只采用符号型数据作为以后所介绍命令的参数。

3.1.1 字符型数据变量的创建

在 MatLab 的工作空间内，字符型数据变量同数值型变量一样是以矩阵形式进行保存的。它的创建方法为：

```
Var='expression'
```

例 1

```
Alfa='Alfa'
V='a*b+c*d'
S='I am a man.'
N='1+sqrt(5))/2'
```

以上结构显示为：

```
Alfa =
Alfa
V =
a*b+c*d
S =
I am a man.
N =
1+sqrt(5))/2
```

此时可检查一下前面 4 个字符变量的大小：

```
SAlfa=size(Alfa);
SV=size(V);
```

```
SS=size(S);
SN=size(N);
```

结果为：

```
SA =
     1     4
SV =
     1     7
SS =
     1    11
SN =
     1    12
```

例 1 的结果充分表明了字符型变量是以矩阵的形式存储在 MatLab 的工作空间内的。

3.1.2　符号型数据变量的创建

创建符号型数据变量需要专门的命令 sym 和 syms。

sym 命令的用处之一是创建单个的符号变量，创建方法如下：

例 2

```
A=sym('A')
Alfa2=sym('Alfa2')
x=sym('x')
```

结果显示为：

```
A =
A
Alfa2 =
Alfa2
x =
x
```

此时可检查一下前面 3 个字符变量的大小：

```
SA=size(A)
Salfa2=size(Alfa2)
Sx=size(x)
```

结果为：

```
SA =
     1     1
Salfa2 =
     1     1
Sx =
1     1
```

由例 2 可知，符号变量在工作空间内的保存是以不同于矩阵形式的单独形式保存的。

syms 命令的使用则要比 sym 方便，它一次可以创建任意多个符号变量，而且命令的格式方程简练。因此一般以 syms 命令来创建符号变量。它的使用格式为：

```
syms var1 var2 ....
```

例 3

```
syms Alfa3 u w v
```

该命令执行后，屏幕并无任何反应，但这 4 个符号变量已存在于 MatLab 的工作空间了。此时可用 whos 命令检查存在于工作空间的各种变量及其所属类型。结果显示为：

```
        whos
Name          Size          Bytes  Class

A             1x1             126  sym object
Alfa          1x4               8  char array
Alfa2         1x1             134  sym object
N             1x12             24  char array
S             1x11             22  char array
SA            1x2              16  double array
SAlfa         1x2              16  double array
SN            1x2              16  double array
SS            1x2              16  double array
SV            1x2              16  double array
Salfa2        1x2              16  double array
Sx            1x2              16  double array
V             1x7              14  char array
x             1x1             126  sym object

Grand total is 58 elements using 566 bytes
```

3.2 符号表达式与符号方程的创建

创建符号表达式和符号方程的目的就是将表达式和方程赋值给一个变量，这个变量也就成了符号变量。而引入这个符号变量后，再引用相应的表达式和方程就方便了许多，不必再一个个重新输入了。

凡是用到 sym 命令的时候，由于在 sym 命令内，表达式和方程式都对空格是敏感的，因此，不要随意添加空格符到式中，以免影响以后的运算结果。

3.2.1 符号表达式的创建

经常使用的符号表达式的创建方法有两种，它们各有自己的优点和缺陷，因此需要根据不同的使用场合选择使用。下面分别介绍这两种创建方法。

1. 用 sym 命令直接创建符号表达式

这种创建方式不需在前面有任何说明，因此使用非常快捷。但在此创建过程中，包含在表达式内的符号变量并未得到说明，也就不存在于工作空间。

例 4

(在运行下列命令前可使用 clear 命令将工作空间清空，以便保证程序顺利运行)

```
f=sym('a*x^2+b*x+c')
f-a
```

运行结果为：

```
f =
a*x^2+b*x+c
??? Undefined function or variable 'a'.
```

例 4 清楚地说明虽然符号表达式 a*x^2+b*x+c 创建成功并将其赋予了变量 f，但表达式所包含的符号变量 a，b，c，x 并未得到说明或创建，因而系统不能识别单个的符号变量 a，不能进行 f-a 的运算。

2. 按照普通书写形式创建符号表达式

这种创建方法与 sym 命令相反。它需要在具体创建一个符号表达式之前，就将这个表达式所包含的全部符号变量创建完毕。但在创建这个表达式时，只需按给其赋值时的格式输入即可完成。

例 5

(在运行下列命令前可使用 clear 命令将工作空间清空，以便保证程序顺利运行)

```
syms a b c x
    f=a*x^2+b*x+c
    f-a
```

运行结果为：

```
f =
a*x^2+b*x+c
ans =
a*x^2+b*x+c-a
```

3.2.2　符号方程的创建

符号方程与符号表达式不同，表达式只是一个由数字和变量组成的代数式，而方程则是由表达式和等号组成的等式。在 MatLab 中，符号方程的创建方法类似于创建符号表达式的第 1 种方法，因为在格式上，创建符号方程不能同创建符号表达式的第 2 种方法一样有如下的命令：

```
ef=a*x^2+b*x+c=0
```

创建符号方程的唯一方法是：

```
equ=sym('EQUATION')
```

例 6

```
e1=sym('a*x^2+b*x+c=0');
e2=sym('x*y*z=e');
```

3.3 符号矩阵的创建

符号矩阵的创建方法要比创建符号表达式的方法多一些，下面分别进行介绍：

3.3.1 用 sym 命令直接创建符号矩阵

这时 sym 命令的使用方法与前面创建符号表达式及方程的用法类似。所创建的符号矩阵的元素可以是任何符号变量及符号表达式和方程，且元素的长度允许不同。在输入格式上，矩阵行之间以“；”隔断，各矩阵元素之间用“，”或空格分隔。

例 7

```
stranger=sym('[1 x/0 sin(x);y/x, 1+1/y, tan(x/y)=0;1=0 3+3, 4*r]')
stranger =
[          1,         x/0,      sin(x)]
[        y/x,       1+1/y,  tan(x/y)=0]
[        1=0,         3+3,         4*r]
```

矩阵 stranger 的确表现出了 sym 命令对所创建矩阵的元素不加限制。但在例 7 中“，”与空格同用只是为了表现在分隔元素上二者作用的等同，在实际使用中，为了格式与页面的整洁，建议只采取一种分隔方法。

3.3.2 以类似创建普通数值矩阵的方法创建符号矩阵

这种创建方法与按照普通书写形式创建符号表达式的方法类似，同样需要在创建符号矩阵之前，就将这个矩阵的元素所包含的全部符号变量创建完毕。而在创建这个矩阵时，只需按创建普通数值矩阵的格式输入即可完成。

例 8

```
syms x y z a b c
f=a*x^2+b*x+c;
g=x*y*z;
h=(f+g)*b/a;
e1=sym('a*x^2+b*x+c=0');
e2=sym('x*y*z=0');
M=[1 2 3 x
   f g h y
e1 e2 e3 z]
```

结果显示为：

```
M =
[        1,                  2,                3,                x]
[   a*x^2+b*x+c,    x*y*z,   (a*x^2+b*x+c+x*y*z)*b/a,           y]
[   a*x^2+b*x+c=0,                   x*y*z=0,     h=0,           z]
```

3.3.3 由数值矩阵转换为符号矩阵

由于数值型和符号型是 MatLab 的两种不同数据类型，因此在 MatLab 中，分属于这两个数据类型的变量之间不能直接运算，而是在 MatLab 的工作空间内将数值型变量转换为符号型变量后进行计算。这个转化过程是在系统内部自动完成的，也可通过命令将数值量转化为符号量，并将这个新产生的符号量赋值给另一变量，以利于后面的计算。

将一个数值矩阵 M 转化为符号矩阵 S 的命令为：

```
S=sym(M)
```

例 9

```
M=[1 2 3 5;7 11 13 17;19 23 29 31]
S=sym(M)
```

结果显示为：

```
M =
     1     2     3     5
     7    11    13    17
    19    23    29    31

S =
[  1,   2,   3,   5]
[  7,  11,  13,  17]
[ 19,  23,  29,  31]
```

说明：不管原来数值矩阵 M 是以分数还是浮点数形式赋值的，但当它被转化为符号矩阵后，都将以最接近原数的精确有理形式给出。

例 10

```
N=[1/3 0.333 0.3333;2^0.5 1.14 1.141;log(3) 1.098 1/0.7]
R=sym(N)
```

结果显示为：

```
N =
      0.33333        0.333        0.3333
       1.4142         1.14         1.141
       1.0986        1.098        1.4286

R =
[  1/3,                 333/1000,      3333/10000]
```

```
[  sqrt(2),            57/50,          1141/1000]
[ 4947709893870347*2^(-52), 549/500,     10/7]
```

3.3.4 利用矩阵元素的通式创建符号矩阵

创建一个如下形式的矩阵 M：

```
M =
[      1/(1+a),     1/(4+a^2),     1/(9+a^3),     1/(16+a^4)]
[   1/(25+a^5),    1/(36+a^6),    1/(49+a^7),     1/(64+a^8)]
[   1/(81+a^9),  1/(100+a^10),  1/(121+a^11),  1/(144+a^12)]
[ 1/(169+a^13),  1/(196+a^14),  1/(225+a^15),  1/(256+a^16)]
[ 1/(289+a^17),  1/(324+a^18),  1/(361+a^19),  1/(400+a^20)]
[ 1/(441+a^21),  1/(484+a^22),  1/(529+a^23),  1/(576+a^24)]
```

如果一项一项的输入，太繁琐了。而此矩阵 M 还是有些规律的，处于第 r 行第 c 列的元素为：

```
M(r, c)= 1/((4*r-4+c)^2+a^(4*r-4+c))
```

可以利用这个规律。在以前版本的 MatLab 中，只要在 sym 命令中加上几个相应的参数就能完成指令，在新版本里没有 sym 命令的这个利用矩阵元素的通式创建符号矩阵的功能。不过可以自创一个函数来实现这个指令：

```
function M=symmat(row, column, f)
%
% SYMMAT 命令是利用通式来创建符号矩阵
% symmat(row, column, f)参数 row, column 分别
% 是待创建符号矩阵的行数和列数
% f 则为矩阵元素的通式

for R=1: row
   for C=1: column
      c=sym(C);
      r=sym(R);
      M(R, C)=subs(sym(f));
   end
end
```

在这个函数中，以“%”提示的内容是本函数的说明和帮助部分。通过这几行文字，可以知道该命令所需的参数及其含义，而且可以用 help 命令来单独查阅该命令的说明信息。

例 11

试创建以下 3 个矩阵：

```
A =
[ sin(1),  sin(2),  sin(3)]
[ sin(4),  sin(5),  sin(6)]
[ sin(7),  sin(8),  sin(9)]
```

```
B =
[  exp(1),   exp(5),   exp(9)]
[  exp(2),   exp(6),  exp(10)]
[  exp(3),   exp(7),  exp(11)]
[  exp(4),   exp(8),  exp(12)]

C =
[      x+y,    2*x+5*y,    3*x+9*y,    4*x+13*y,    5*x+17*y]
[   4*x+2*y,    5*x+6*y,   6*x+10*y,    7*x+14*y,    8*x+18*y]
[   7*x+3*y,    8*x+7*y,   9*x+11*y,   10*x+15*y,   11*x+19*y]
[  10*x+4*y,   11*x+8*y,  12*x+12*y,   13*x+16*y,   14*x+20*y]
[  13*x+5*y,   14*x+9*y,  15*x+13*y,   16*x+17*y,   17*x+21*y]
```

具体命令为：

```
syms x y c r
a=sin(c+(r-1)*3;
b=exp(r+(c-1)*4;
c=(c+(r-1)*3)*x+(r+(c-1)*4)*y;

A=symmat(3, 3, a);
B=symmat(4, 3, b);
C=symmat(5, 5, c);
```

由于在函数 symmat 中，采用了 M(R, C)=subs(sym(f))的方法，因此当 f 为字符参数时，symmat 命令同样可以给出正确答案。

例 12

```
A=symmat(3, 3, 'sin(c+(r-1)*3)')
B=symmat(4, 3, 'exp(r+(c-1)*4)')
C=symmat(5, 5, '(c+(r-1)*3)*x+(r+(c-1)*4)*y')
```

3.4 创建实数和复数

在 MatLab 中，进行如下运算：

```
syms x y
a=real(x);
b=imag(x);
c=conj(x+y);
```

其结果为：

```
a =
1/2*x+1/2*conj(x)
b =
-1/2*i*(x-conj(x))
```

```
c =
conj(x+y)
```

可见只用 syms x 命令来创建变量，系统将认为所创建的变量 x 是复数。但在实际工程、科学计算中经常需要用到如此表达的复数：z=x+yi (x，y∈R)，这种表达方法的实现就涉及到了在 MatLab 中如何表达一个实数的问题。sym 命令可以完成这个任务，其使用格式为：

```
x=sym(x, 'real');
y=sym(x, 'real');
或：syms x y real
z=x+y*i;
```

通过以上步骤创建的变量 x，y 系统将认为其是实数，变量 z 是一个虚数。此时再用 real，imag，conj 命令进行检查，有：

```
syms x y real
z=x+y*i;
a=real(x)
b=imag(y)
c=conj(z)
```

结果为：

```
a =
x
b =
0
c =
x-i*y
```

当想清除变量 x，y 的实数属性时，需要用以下命令：

```
x=sym(x, 'unreal')
或 syms x unreal
```

注意：只有上面两个命令才能清除变量的实数属性，如果只是简单地用清除命令 clear x 将变量 x 从工作空间中清除，则当下次再创建符号变量 x 时，变量 x 仍将保持实数的性质。

3.5 数值变量、符号变量与字符变量的相互转换

在 MatLab 的工作空间中，数值、符号和字符是 3 种主要的基本数据类型。它们之间的等级是不一样的，数值变量最低，符号变量最高，字符变量居中。如有这 3 种变量的混合运算，则系统要先将参与运算的所有变量自动统一转换成其中变量等级最高的类型，然后再进行计算。也可通过命令来完成不同类型数据间的转换，此处用到的命令较多，按不

同的目标转换类型，可分为 3 类：

3.5.1 转换为数值变量

1. x=double(S)

当 S 为符号变量时，此命令将 S 转换为数值变量 x。如 S 中含有非数字的符号，则系统将给出错误提示。

当 S 为字符变量时，此命令将 S 转换为数值矩阵 x。矩阵中元素的值为 S 中相应字符的 ASIIC 值。

例 13

```
S1=sym(23.4);
x1=double(S1)
S2=sym('32*a');
x2=double(S2)
S3='45.323';
x3=double(S3)
S4='23/56';
x4=double(S4)
S5=['238';'2df';'2a3';'*)4'];
x5=double(S5)
```

转换结果为：

```
x1 =
                    23.4
x3 =
    52    53    46    51    50    51
x4 =
    50    51    47    53    54
x5 =
    50    51    56
    50   100   102
    50    97    51
    42    41    52
```

至于 x2 则有以下错误提示：

```
??? Undefined function or variable 'a'.
Error in ==> E: \toolbox\symbolic\@sym\double.m
On line 45  ==> D = reshape(eval(X), m, n);
Error in ==> E: \bin\book\temp31.m
On line 4  ==> x2=double(S2)
```

2. x=str2num(S)

str2num 是专门用来将字符变量转换为数值变量 x 的命令。但当 S 是一个包含非数字

字符的变量时，str2num(S)命令将返回一个空矩阵[]。

例 14

```
S1='3.456';
x1=str2num(S1)
S2='3/23';
x2=str2num(S2)
S3=['238';'223';'243';'124'];
x3=str2num(S3)
S4='23e';
x4=str2num(S4)
```

转换结果为：

```
x1 =
                    3.456
x2 =

        0.130434782608696
x3 =
   238
   223
   243
   124
x4 =
     []
```

3. x=numeric(S)

命令 numeric(S) 将变量 S 转换为数值量 x，不管 S 是字符变量还是符号变量。但 S 不能是矩阵，否则将给出错误提示。

例 15

```
S1=sym('3/23');
x1=numeric(S1)
S2='3.456';
x2=numeric(S2)
S3=['238';'223';'243';'124'];
x3=numeric(S3)
S4='23f';
x4=numeric(S4)
```

转换结果为：

```
x1 =
        0.130434782608696
x2 =
                    3.456
```

而 x3，x4 将给出错误信息。

3.5.2 转化为符号变量

命令为：S=sym(f) 此命令对变量 f 不作类型限制，只要不是非法的表达形式或字符矩阵，sym(f)即可将其转换为符号变量 S。

例 16

```
S1=sym('3.456')
S2=sym(3.456)
f3='23f';
S3=sym(f3)
f4=['23';'23';'32';'23'];
S4=sym(f4)
```

转换结果为：

```
S1 =
3.456
S2 =
432/125
```

而 S3，S4 将给出错误信息。

3.5.3 转换为字符变量

1. s=int2str(x)

此命令将整数 x 转换为字符变量 s。当 x 是普通有理数时，将对 x 四舍五入后进行转换。当 x 为虚数时，将只对其实部进行转换。

例 17

```
x1=-23;
s1=int2str(x1)
x2=-2.7;
s2=int2str(x2)
x3=12.9-4*i;
s3=int2str(x3)
```

转换结果为：

```
s1 =
-23
s2 =
-3
s3 =
13
```

2. s=num2str(x)

此命令将普通数值变量 x 转换为字符变量 s。在 int2str 命令中对 x 的限制则全部取消。
例 18

```
x1=-23;
s1=num2str(x1)
x2=-2.7;
s2=num2str(x2)
x3=12.9-4*i;
s3=num2str(x3)
```

转换结果为：

```
s1 =
-23
s2 =
-2.7
s3 =
12.9-4i
```

3.6 基本画图功能——函数二维图形的表达

MatLab 的数据可视化功能具有若干个专门工具箱，进行从数据到图像的处理。将在第 9 章对此进行详细的介绍。本节简单说明绘制一元函数图像的方法。

3.6.1 适用于数值量的二维图形命令 plot

Plot 命令是 MatLab 的内部函数，也是其最基本的图形命令。主要有以下几种使用格式：

1. plot(Y)

此命令中参数 Y 可以是向量、实数阵和复数阵。
绘图时，以 Y 每列元素的相应下标值为横坐标，以 Y 的元素为纵坐标制得连线图。
例 19

```
Y= peaks;
plot(Y)
```

结果如图 3.1 所示。

2. s=num2str(x)

此命令将普通数值变量 x 转换为字符变量 s。在 int2str 命令中对 x 的限制则全部取消。
例 18

```
x1=-23;
s1=num2str(x1)
x2=-2.7;
s2=num2str(x2)
x3=12.9-4*i;
s3=num2str(x3)
```

转换结果为：

```
s1 =
-23
s2 =
-2.7
s3 =
12.9-4i
```

3.6 基本画图功能——函数二维图形的表达

MatLab 的数据可视化功能具有若干个专门工具箱，进行从数据到图像的处理。将在第 9 章对此进行详细的介绍。本节简单说明绘制一元函数图像的方法。

3.6.1 适用于数值量的二维图形命令 plot

Plot 命令是 MatLab 的内部函数，也是其最基本的图形命令。主要有以下几种使用格式：

1. plot(Y)

此命令中参数 Y 可以是向量、实数阵和复数阵。
绘图时，以 Y 每列元素的相应下标值为横坐标，以 Y 的元素为纵坐标制得连线图。
例 19

```
Y= peaks;
plot(Y)
```

结果如图 3.1 所示。

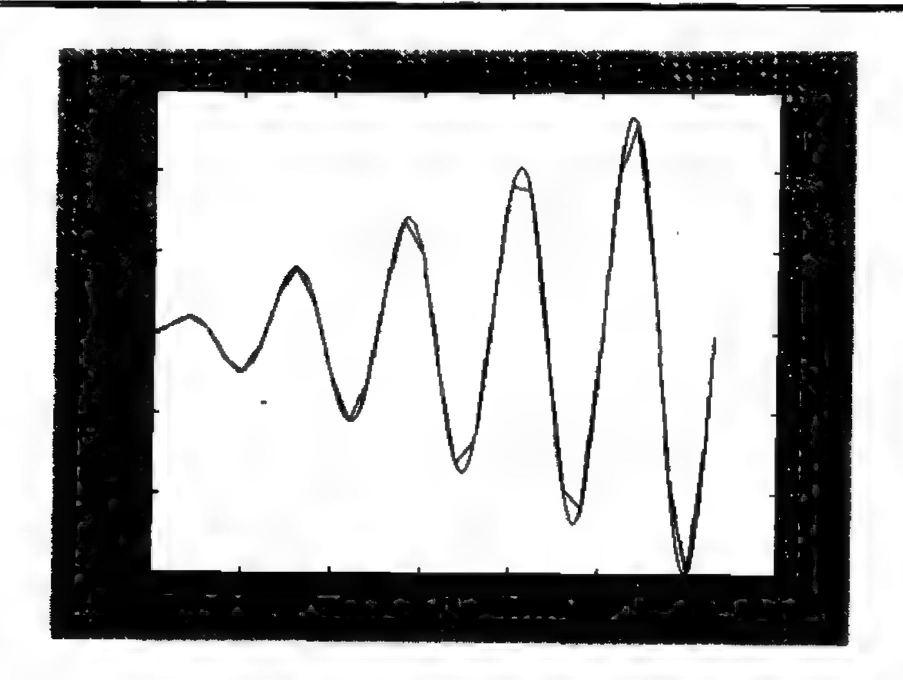


图 3.2　自变量间隔大小的不同对图形的影响

图 3.2 明显表现出由于 X2 比 X1 的采样点多 100 倍，因此由前者绘制得的连线图要比后者光滑得多。

例 21

```
X=0: 0.01: 1.5;
Y=[X.^0.3;X.^0.7;X.^1;X.^2;X.^5];
plot(X, Y)
```

结果如图 3.3 所示。

```
Y=[X.^0.3;X.^0.7;X.^1;X.^2;X.^5];
```

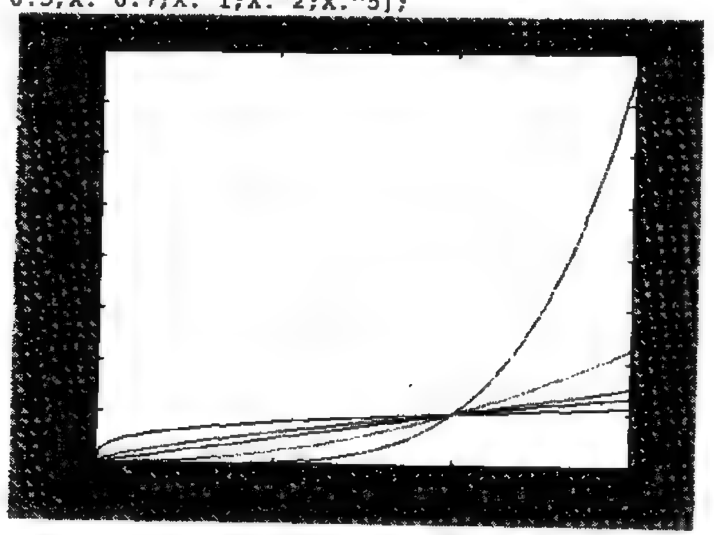


图 3.3　函数 Y=X.^0.3;X.^0.7;X.^1;X.^2;X.^5 在[0,15]区域上的图像

例 22

```
A=0: 0.01: 2;
X=[A;sin(A);A*2;A.^2]';
Y=[sin(A);A;exp(A);cos(A)]';
plot(X, Y)
```

结果如图 3.4 所示。

```
Y=[sin(A);A;exp(A);cos(A)]'
```

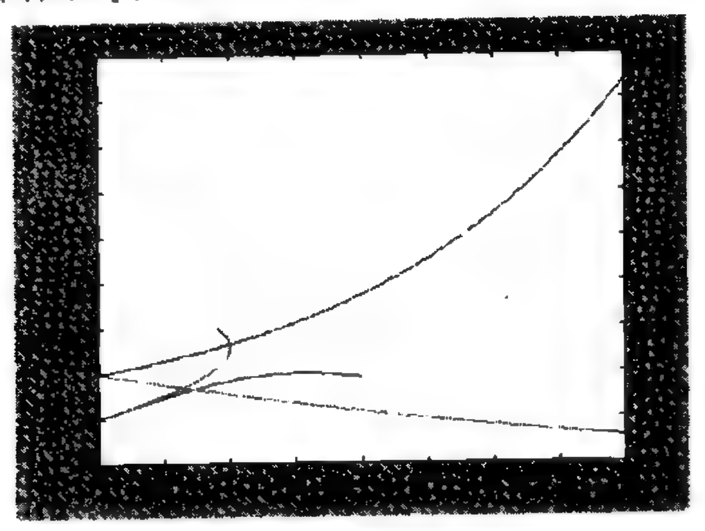


图 3.4　横纵坐标均可控制的 4 个不同函数的图形

此命令将每一对 X，Y(即 X1 与 Y1，X2 与 Y2，…)按照 plot(X，Y)的格式在同一幅图内绘得若干条颜色不同的连线。因此对每一对 X，Y 的要求就是必须符合命令 plot(X，Y)的要求。可试一试将例 22 中的两个 plot 命令改写为 1 个 plot 命令。

3.6.2　专门用于绘制一元函数曲线的命令 fplot

在 plot 命令中，系统是将从外部输入或间接确认的数值矩阵转化为连线图的。而在实际应用中绘制函数的二维曲线时，一般并不清楚函数的具体情况，因而在确定自变量 x 的取值间隔时，往往一律用平均间隔，这样往往不大准确。取好了，还可以表现出函数图像的大概情况，取差了，则会因某处 X 元素的间隔太大而根本错绘了曲线，不能反映出函数的变化情况从而使绘图失败。

fplot 命令正是针对这个问题产生的，它用于指导绘图的数值点矩阵，不是直接从外部接收，而是通过其内部自适应算法而产生的。即在函数值变化比较平稳处，它所取的数值点就会自动相对稀疏一些，在函数值变化剧烈处，所取的数值点就会自动密集一些。所以对于曲线起伏剧烈(也就是函数的二阶导数很大)的函数，用 fplot 命令将比用一般等间距取点的 plot 命令绘得的曲线光滑准确一些。

fplot 命令的具体使用格式为：

```
fplot('function_name', limits, tol)
```

```
[x, y]=fplot('function_name', limits, tol)
```

上式中各项参数的含义为：

function_name　待绘制函数曲线的函数的名称。要求函数必须是 MatLab 已有的函数，或是自定义的 m 函数。可以是一个向量函数。

limits　limits=[x1，x2]，为 x 的取值空间。

x，y　fplot 命令计算得的数据点的坐标。如采用 fplot('function_name'，limits)的格式，将直接画出图形。

tol　为 fplot 命令在进行运算中的相对误差。tol 越小，所绘得的曲线就越接近实际曲线的情况，但系统要为此占用很大的资源。

例 23

fplot 与 plot 命令的比较：

(1) 创建函数：

```
function y=funfplot(x)
y=sin(1./tan(pi.*x));
```

(2) 用 fplot 命令求得坐标点，并按照同样大小创建一个等间隔坐标点：

```
[X, Y]=fplot('funfplot', [-0.1, 0.1], 2e-4);
n=size(X);
x=-0.1: 0.2/(n(1)+1): 0.1;
y=funfplot(x);
```

(3) 作图比较 fplot 与 plot 命令：

```
plot(x, y)
figure
plot(X, Y)
```

结果如图 3.5，3.6 所示。

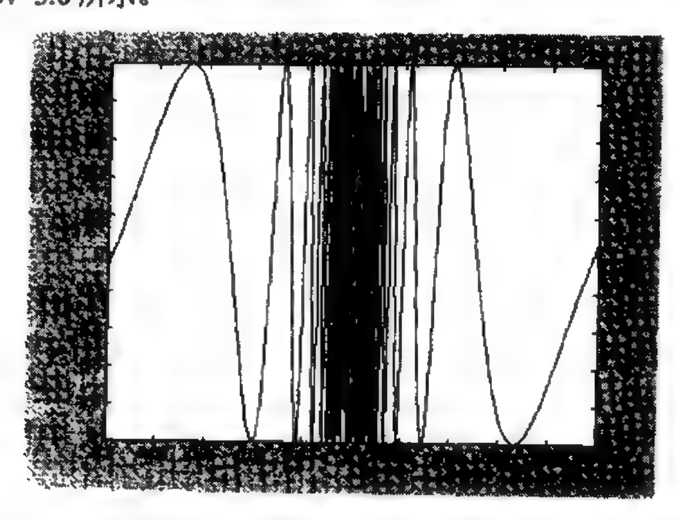


图 3.5　plot 命令绘图结果

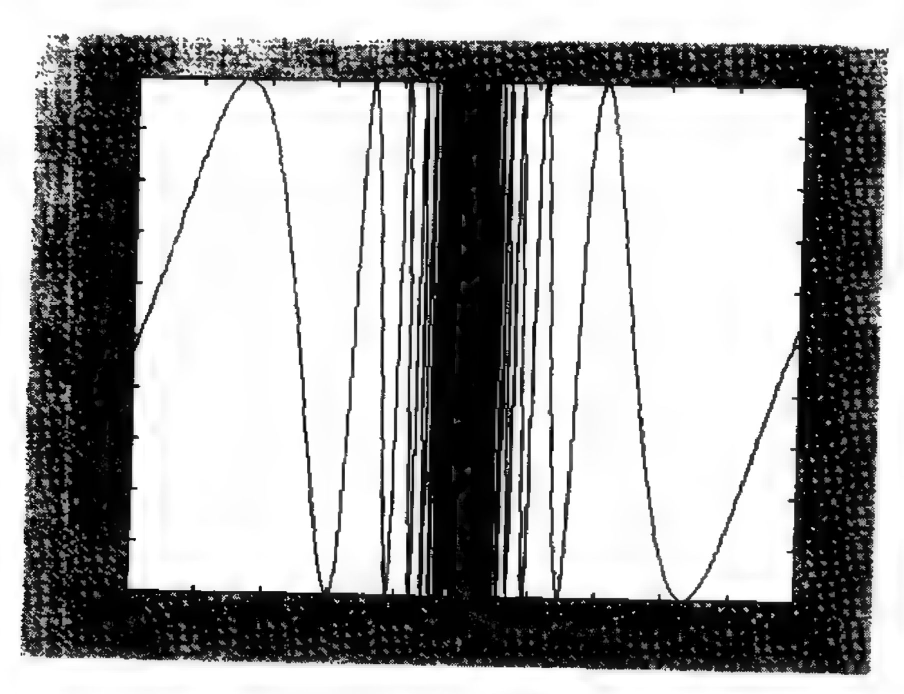


图 3.6　fplot 命令绘图结果

3.6.3　专门用于绘制一元符号函数曲线的命令 ezplot

如果符号函数中只含有一个变量，那么 MatLab 将其视为一个普通函数，画出其在某个区域内的图像。这个命令就是 ezplot，它的使用方法为：

```
ezplot(sym_function, limits)
```

式中参数含义为：

sym_function　符号函数或代表它的符号变量。

Limits　limits=[x1，x2]，为 x 的取值空间。其默认值为[-2pi，2pi]。

例 24

```
syms x
f=(x^2)^(sin(x)^2);
ezplot(f)
```

结果如图 3.7 所示。

```
ezplot(f)
```

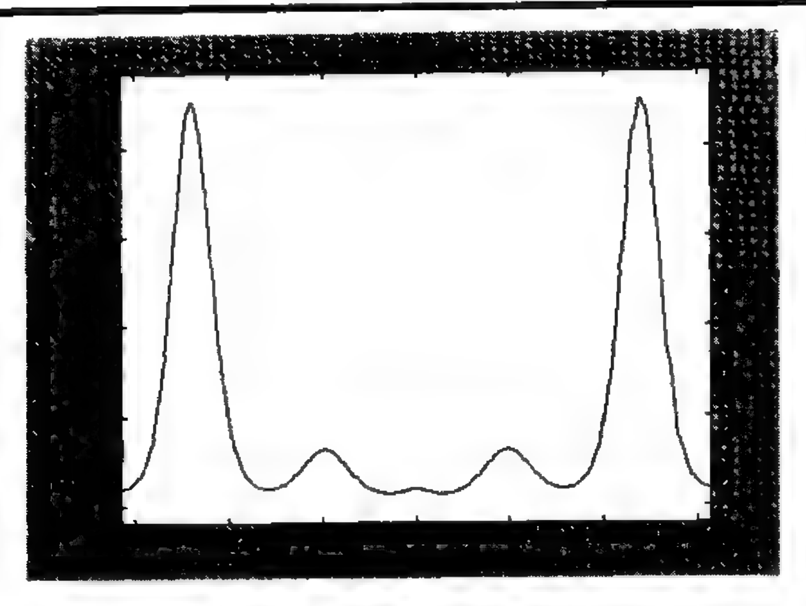


图 3.7 符号函数 f 的图形曲线

3.7 不同精度的运算

在 MatLab 的所有工具箱中的运算，共有 3 种针对不同计算精度的算法，分别为：

- 针对浮点运算的数值算法。
- 针对精确运算的符号算法。
- 可控精度的算法。

在以上 3 种不同的算法中，针对浮点运算的数值算法是计算速度最快的算法，它与 C，FORTURN 语言中的浮点运算算法完全相同。由于其在机器内部都是以若干位二进制数表达，因此它在机器内的表达和计算结果都是一个被“截断”的近似值。

例 25

在 MatLab 中计算：

```
a=1/2+3/7;
```

其结果为：

```
a =
        0.928571428571429
```

在计算 a 的过程中，浮点运算共有 3 处产生了误差：第 1 处在计算 1/2 处；第 2 处在计算 3/7 处；第 3 处在计算两者的和处。

与浮点运算的算法不同，第 2 种算法——针对精确运算的符号算法是以运算的准确为第 1 要义的算法，它完全是按照有理数的运算方法进行计算。但它要比另外两种算法占用系统更多的时间和资源。

例 26

在 MatLab 中，1/2+3/7 的符号计算为：

```
b=sym(1/2+3/7);
```

其结果为：

```
b =
13/14
```

从结果 b=13/14 可以看出，符号运算中没有产生误差。

可控精度的算法并不是独立于以上两种算法的新算法，它可以说是以上两种算法的折衷。在 MatLab 中，这种算法可以用命令来控制参加运算的各个量取多少位有效数字进行计算，从而达到控制运算精度的目的。其所取有效数字的多少对运算速度有着重要的影响。在可控精度的运算中，用于控制精度和计算的命令分别为 Digits 和 vpa。

Digits 命令近似于一个全局变量，其作用是控制取多少位有效数字进行计算。它的默认值为 32，在此 32 位有效数字精度下的计算结果大概与浮点运算的精度相同。它有以下几种使用格式：

(1) Digits(n)

(2) Digits(& digits)

第 1 种使用格式是用来控制 MatLab 在其后的可控精度运算中取 n 位有效数字进行计算。第 2 种使用格式中虽然有大小写之分，但都是用来查询 Digits 的值。需要注意的是，如果已经在工作空间创建了变量 digits(如 digits=68)，则变量 digits 将成为自定义的一个普通变量，不再与可控精度计算有任何联系。同样，如已经创建了名为 Digits 的变量(如 Digits=43)，则变量 Digits 亦将成为普通变量而不再与可控精度计算有任何的联系。此时如再需查看系统运算所取的精度时，就没有变量名可用了。

另外，如果使用的是高版本的 MatLab 如 5.0 以上，则在 digits 的帮助文件中，其命令格式全部错写为：

(1) digits=n

(2) digits(n)

vpa 命令的作用是在计算精度为 Digits=n 的情况下进行可控精度运算。其使用格式为：

```
vpa(expression)
```

例 27

```
Digits(5);
A=sym(2/7);
B=sym(1.45224);
c=pi;
d='2.83';
f=3/11;
ans1=vpa(A+B)
ans2=vpa(A+c)
ans3=vpa(c+d)
```

```
ans4=vpa(c+f)
Digits(40);
ans5=vpa(A+B)
ans6=vpa(A+c)
ans7=vpa(c+d)
ans8=vpa(c+f)
```

结果为：

```
ans1 =
1.7380
ans2 =
3.4273
ans3 =
5.9716
ans4 =
3.4143
ans5 =
1.737954285714285714285714285714285714286
ans6 =
3.427306939304078952748357668993788598483
ans7 =
5.971592653589793187052237044554203748703
ans8 =
3.414319926317065601040212641237303614616
```

注意：在 vpa 的表达式中，不管参与运算的变量是哪种类型，其计算结果均是符号量。

3.8 创建抽象函数

在实际的工程和科学计算中，经常用 f(x)，g(x，y，z)或 f，g 这样的符号来代表一个或未知或已知的函数进行运算。这种功能在 MatLab 中既可以用普通的符号命令 sym 来完成，也可用 MAPLE 函数库中的 map 命令完成。

3.8.1 用 sym 命令创建抽象函数

如果函数 f 是变量 var1，var2，…，varn 的抽象函数，则其创建方法为：

```
syms var1 var2 ... varn
f=sym('f(var1, var2, ..., varn')
```

例 28

```
syms x y z
f=sym('f(x, y, z)')
```

结果为：

```
f =
f(x, y, z)
```

变量 vari 不仅可以是单一的独立变量，还可以是其他变量的计算表达式。

例 29

```
g=sym('g(x, x+y, x*y*z, sin(x*y+z))')
```

结果为：

```
g =
g(x, x+y, x*y*z, sin(x*y+z))
```

用此方法创建出来的抽象函数完全可以像普通函数一样进行加减乘除、求导积分等运算。

例 30　对例 29 创建的抽象函数 g 进行关于变量 x 的求导计算

```
diff(g, x)
```

结果为：

```
ans =
D[1](g)(x , x+y , x*y*z , sin(x*y+z))+D[2](g)(x , x+y , x*y*z ,
sin(x*y+z))+D[3](g)(x,x+y,x*y*z,sin(x*y+z))*y*z+D[4](g)(x,x+y,x*y*z,
sin(x*y+z))*cos(x*y+z)*y
```

3.8.2　用 map 命令创建抽象函数

map 命令用起来方便灵活，它的主要作用是对一系列变量进行一定的操作。它的使用格式为：

```
map(fcn, expr, arg2, ..., argn)
map2(fcn, arg1, expr, arg3, ..., argn)
```

上式中各个参数的含义为：

fcn　一个操作手续或名称

expr　表达式

argi　用于操作的对象

例 31

```
A=maple('map(f, x, y, z)')
B=maple('map(f, x+y, z)')
C=maple('map(f, x*y, z)')
D=maple('map(f , {x, y, z})')
E=maple('map(x -> x^2, x + y)')
```

结果为：

```
A =
f(x, y, z)
B =
f(x, z)+f(y, z)
C =
f(x, z)*f(y, z)
D =
[f(x), f(y), f(z)]
E =
x^2+y^2
```

例 31 创建出的抽象函数 A，B，C，D，E 均与 sym 命令中创建的函数一样，具有相同的性质。

3.9 使用 maple 命令

MatLab 中的符号运算功能都来自 MAPLE V，在本书中介绍的所有符号运算命令只不过是其中一小部分常用命令。如果想深入研究 MAPLE V，则需要运用 maple 命令。

maple 命令的参数很多，使用也很灵活，在此介绍 2 个最常用的使用格式：

3.9.1 maple(statement)

此命令的作用是将对变量、表达式、函数等的描述(statement)传递到 MatLab 符号运算的引擎——MAPLE V，由 MAPLE 进行计算并返回字符结果。通过这个命令格式，可以调用 MAPLE 函数库中的非图像处理函数的所有函数，因此这个命令十分重要，需认真掌握。

例 32

s=evalf(num，n)命令是 MAPLE 中类似 vpa 的一个命令，它的作用是将数值 num 以 n 位数字表达出来

```
ans1=maple('evalf(pi, 10)')
ans2=maple('evalf(0.5, 10)')
ans3=maple('evalf(0.5000000100000003, 10)')
ans4=maple('evalf(1/2, 10)')
```

结果为：

```
ans1 =
3.141592654
ans2 =
0.5
ans3 =
0.5000000100
```

```
ans4 =
0.5000000000
```

3.9.2 maple('function', arg1, arg2, ···)

这个命令格式是 MatLab 调用 MAPLE 函数库中的函数 function 的标准命令。式中参数 function 是函数的名称，arg1，arg2，…是函数 function 所需的参数。

例 33

将例 32 的命令调用形式改写为：maple('function', arg1, arg2, …)

```
f=1.2;
s='4/3';
S=sym('112.23231442*321');
ansf=maple('evalf', f, 10)
anss=maple('evalf', s, 4)
ansS=maple('evalf', S, 6)
```

运算结果为：

```
ansf =
1.200000000
anss =
1.333
ansS =
36026.6
```

可见此种格式的运算对有些函数参数的要求要宽松一些。

另外，MAPLE 函数库中的函数并未在启动 MatLab 时全部调入内存。所以，对于有些函数在调用之前要先将其调入内存，这就需要先运行 MAPLE 的读库函数(Readlib-defined function)readlib 命令将所需命令读入内存。

例 34 求 x 的 5 阶贝努立表达式

在 MatLab 中，并没有产生贝努立表达式的命令，产生此表达式的命令(bernoulli)存在于 MAPLE 的函数库中。当直接用 maple 调用 bernoulli 命令时，系统有如下反应：

```
syms x
maple('bernoulli', 5, x)
???Undefined function or variable 'bernoulli'.
```

上面的提示说明 bernoulli 函数尚未被调入。其调入命令为：

```
maple('readlib(bernoulli)')
```

按回车键后，得到其版权和函数信息 ans：

```
ans =
proc (n, x) local t, b, i; options remember, 'Copyright (c) 1990 by
the University of Waterloo. All rights reserved.';
..................
```

调入 bernoulli 命令以后，就可以用 maple 命令调用执行了：

```
maple('bernoulli', 5, x)
```

结果为：

```
ans =
-1/6*x+5/3*x^3+x^5-5/2*x^4
```

当 maple 的函数调用的参数为一数值矩阵时，maple 将给出类似如下的结果：

```
x=[3 5;4 7];
maple('bernoulli', 5, x)

ans =
-1/6*MATRIX([[3 ,   5] ,   [4 ,   7]])+5/3*MATRIX([[3 ,   5] ,   [4 ,
7]])^3+MATRIX([[3,  5],  [4,  7]])^5-5/2*MATRIX([[3,  5],  [4,  7]])^4
```

注意：格式 2 虽然是调用 MAPLE 函数库中函数的标准格式，但由于其对格式有较严格的限制，因此在对一些函数的调用中，系统虽然承认该函数名，却并不对其进行计算。所以，在一般使用过程中，建议选用第 1 种使用格式。

附注：在老版本的 MatLab 和 MAPLE 中，还有一个命令为 mpa，在 MatLab 5.2 中已经不用了。

3.10 MAPLE V 中的特殊函数及其使用方法

除了MatLab提供的50余个特殊函数外，在MAPLE V的函数库中，还有40余个不同的特殊函数可供使用。这些函数将MatLab在计算特殊函数的功能上又提高了一大步。借助MAPLE V这个“强力引擎”，可以求得菲涅耳余弦积分(Fresnel Cosine Integral)、完成在复数平面上的误差函数(Error Function)的计算等复杂的计算。

这些特殊函数及其所需参数如下表所示：

```
bernoulli        n        Bernoulli Numbers
bernoulli        n, z     Bernoulli Polynomials
BesselI          x1, x    Bessel Function of the First Kind
BesselJ          x1, x    Bessel Function of the First Kind
BesselK          x1, x    Bessel Function of the Second Kind
BesselY          x1, x    Bessel Function of the Second Kind
Beta             z1, z2   Beta Function
binomial         x1, x2   Binomial Coefficients
LegendreKc       x        Complete Elliptic Integral of First Kind
LegendreEc       x        Complete Elliptic Integral of Second Kind
LegendrePic x1, x         Complete Elliptic Integral of Third Kind
LegendreKc1      x        LegendreKc using Complementary Modulus
LegendreEc1 x             LegendreEc using Complementary Modulus
LegendrePic1     x1, x    LegendrePic using Complementary Modulus
```

```
    erfc          z         Complementary Error Function
    erfc          n, z      Complementary  Error  Function's  Iterated
Integrals
    Ci            z         Cosine Integral
    dawson        x         Dawson's Integral
    Psi           z         Digamma Function
    dilog         x         Dilogarithm Integral
    erf           z         Error Function
    euler         n         Euler Numbers
    euler         n, z      Euler Polynomials
    Ei            x         Exponential Integral
    Ei            n, z      Exponential Integral
    FresnelC      x         Fresnel Cosine Integral
    FresnelS      x         Fresnel Sine Integral
    GAMMA         z         Gamma Function
    harmonic      n         Harmonic Function
    Chi           z         Hyperbolic Cosine Integral
    Shi           z         Hyperbolic Sine Integral
    hypergeom     X1, X2    (Generalized) Hypergeometric Function
    LegendreF     x, x1     Incomplete Elliptic Integral of First Kind
    LegendreE     x, x1     Incomplete Elliptic Integral of Second Kind
    LegendrePi    x, x2, x1     Incomplete Elliptic Integral of Third Kind
    GAMMA         z1, z2        Incomplete Gamma Function
    W             z             Lambert's W Function
    W             n, z          Lambert's W Function
    lnGAMMA       z             Logarithm of the Gamma function
    Li            x             Logarithmic Integral
    Psi           n, z          Polygamma Function
    Ssi           z             Shifted Sine Integral
    Si            z             Sine Integral
    Zeta          z             (Riemann) Zeta Function
    Zeta          n, z          (Riemann) Zeta Function
    Zeta          n, z, x       (Riemann) Zeta Function

      Orthogonal Polynomials (Extended Symbolic Math Toolbox only)
    T             n, x          Chebyshev of the First Kind
    U             n, x          Chebyshev of the Second Kind
    G             n, x1, x      Gegenbauer
    H             n, x          Hermite
    P             n, x1, x2, x Jacobi
    L             n, x          Laguerre
    L             n, x1, x      Generalized Laguerre
    P             n, x          Legendre
```

在上面这些特殊函数的使用中，maple命令就只能调用其中的几个函数，命令mfun则是调用它们的专用命令。

mfun命令用于计算MAPLE函数库中的特殊函数的数值结果。它的使用格式为：

```
mfun('function', p1, p2, ..., pk)
```

式中function是MAPLE库中的特殊函数的函数名。 p1，p2，…，pk为计算特殊函数function时所需的参数。在这些参数中，除了最后一个参数 pk 可以是矩阵外，其他参数则必须严格符合在MAPLE V中调用该函数时相应的数据类型。如果上面的格式简化为mfun('function')，即不给函数任何参数，则命令mfun将返回一个非数NAN。

例 35　试计算以下特殊函数

- 计算菲涅耳余弦积分

```
y1=mfun('FresnelC', 2.5)
```

- 计算 23 阶贝努立式在 x=3，x=5 处的值

```
y2=mfun('bernoulli', 23, [3 5])
```

- 计算双曲余弦积分

```
y3=mfun('Chi', 4)
```

- 计算菲涅耳余弦积分

```
y4=mfun('FresnelS', 4)
```

- 计算整数积分在 x=[1.5，2;2.5 3]4 处的值

```
y5=mfun('Ei', [1.5, 2;2.5 3])
```

结果为：

```
y1 =
        0.457413009641777
y2 =
        96469015            405342139861590
y3 =
        9.81354755882319
y4 =
        0.420515754246928
y5 =
          3.3012854491298         4.95423435600189
          7.0737658945786         9.93383257062542
```

3.11　函数计算器

在 MatLab 中，符号运算工具箱还提供了一个用于考察两个一元函数各自性质及其相关关系的可视化命令 funtool。

当输入如下命令时，系统将产生 3 个新的图形窗口，如图 3.8，3.9 和 3.10 所示。

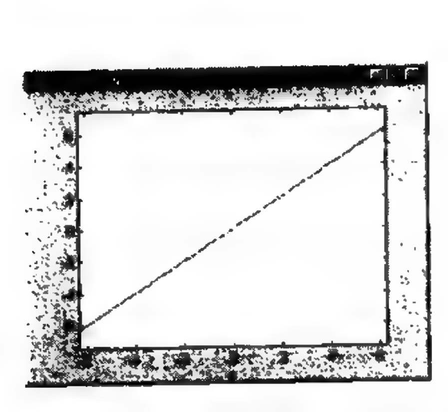


图 3.8　窗口 1

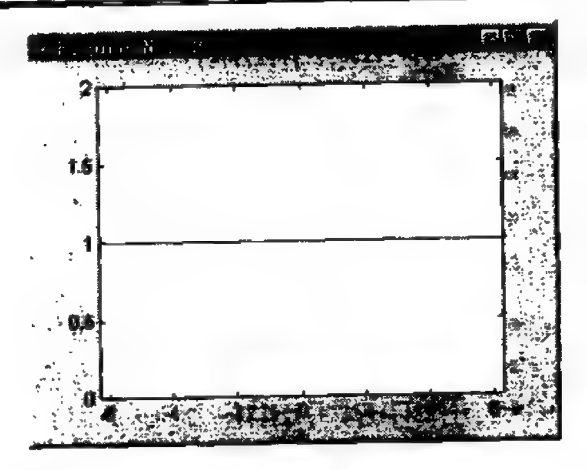


图 3.9　窗口 2

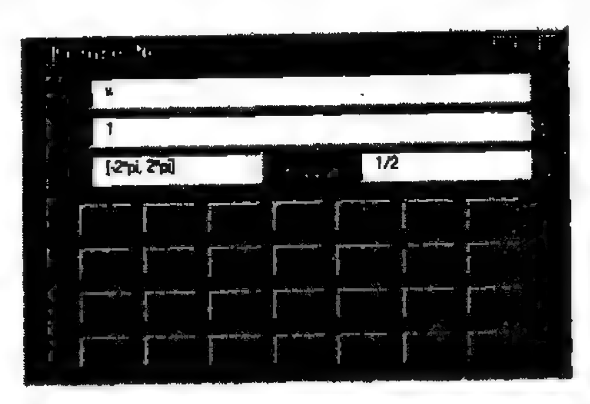


图 3.10　窗口 3

在系统新创建的 3 个图形窗口中，Figure No.3 是控制窗口，在它的上半部分，分别是一元函数 f(x)，g(x)的定义式、自变量 x 的定义域和常数 a 的值，这些都可自设。当完成相应的设置后，图形窗口 Figure No.1 和 Figure No.2 将自动绘出在定义域上函数 f(x)，g(x)的图形。

控制窗口 Figure No.3 的下半部分是运算及帮助部分，共有 3 行。其中第 1 行 7 个按钮的作用分别为：

- 对函数 f(x)求导。
- 对函数 f(x)积分。
- 对函数 f(x)化简。
- 取 f(x)的分子表达式。
- 取 f(x)的分母表达式。
- 求 1/f(x)。
- 求函数 f(x)的反函数。

第 2 行的 7 个按钮的作用为：

- 计算 f(x)+a。
- 计算 f(x)−a。

- 计算 af(x)。
- 计算 f(x)/a。
- 计算 $f^a(x)$。
- 计算 f(x+a)。
- 计算 f(ax)。

第 3 行的 7 个按钮的作用为：

- 计算 f(x)+g(x)。
- 计算 f(x)−g(x)。
- 计算 f(x)g(x)。
- 计算 f(x)/g(x)。
- 计算复合函数 f(g(x))。
- 令 g(x)=f(x)。
- 交换 f(x)和 g(x)的定义式。

最后第 4 行的 7 个按钮的作用为：

- 将当前函数 f(x)送入系统内含的典型函数演示表中。
- 在图形窗口 Figure No.1 中顺序显示典型函数演示表中的函数曲线。
- 将在当前图形窗口 Figure No.1 中的函数 f(x)从典型函数演示表中删除。
- 将函数计算器的状态调整到初始状态：f(x)=x;g(x)=1;x=[−pi，pi]; a=1/2。
- 查看函数计算器的帮助文件。
- 系统自动演示函数计算器的计算功能。
- 关闭函数计算器。

3.12 如何获得符号运算函数及命令的帮助信息

获得符号运算命令和函数的帮助信息有两种不同的方法。第 1 种，查看 MatLab 的附带函数说明文档。可以用网络浏览器如 IE，Netscap 等查阅 HTML 文档，也可以用 Acrobat Reader 阅读浏览器来查看 PDF 文档。其中，HTML 文档的主页 helpdesk.html 保存在 MatLab 所在目录下的 help 子目录内，PDF 文档则保存在 help 的子目录 pdf doc 内。第 2 种方法，就是使用 MatLab 的帮助命令 help。它的使用格式为：

```
help function
```

此式中的 function 是指待查的 MatLab 函数名。

但在查阅符号运算命令的使用方法时，此命令有时会得不到满意的结果。问题在于 MatLab 内部，对于数值量和符号量两类不同数据类型的很多相同运算的命令名是相同的，因此在当调用符号运算命令时，MatLab 将会对其中的有些命令进行重载，即将它们由适用于数值量的运算转换为适用于符号量的运算。而在普通命令 help function 中，命令函数 function 并没有进行运算，自然就没有进行重载，因而由 help 命令查得的帮助信息也就是

只适用于数值量的了。为了能够知道一个命令是否可以在符号运算中应用和查到它的帮助信息，对于所有可进行重载的命令，MatLab 在此命令的适用于数值量运算的帮助信息中，会在所有说明文字结束后，有“重载方法”(Overloaded methods)的提示。可按照提示中的方法查阅到重载后符号运算的帮助信息。

例 36

查阅微分求导命令 diff 在符号运算中的使用方法：

```
help diff
```

查得结果为：

```
DIFF Difference and approximate derivative.
   DIFF(X), for a vector X, is [X(2)-X(1)  X(3)-X(2) ... X(n)-X(n-
   1)].
   DIFF(X), for a matrix X, is the matrix of column differences,
     [X(2: n, : ) - X(1: n-1, : )].
   DIFF(X), for an N-D array X, is the difference along the first
     non-singleton dimension of X.
.........................................
   Examples:
     h = .001; x = 0: h: pi;
     diff(sin(x.^2))/h is an approximation to 2*cos(x.^2).*x
     diff((1: 10).^2) is 3: 2: 19

    See also GRADIENT, SUM, PROD.
Overloaded methods
   help sym/diff.m
help char/diff.m
```

上式中在 Overloaded methods 前面的介绍都是适用于数值量的 diff 命令的使用方法。但最后 3 行介绍的是，diff 命令可以被分别重载后应用于符号量(sym)和字符量(char)。如果想知道 diff 命令关于符号量的用法，则可使用上面提示到的方法：

```
help sym/diff.m
```

查得结果为：

```
DIFF  Differentiate.
 DIFF(S) differentiates a symbolic expression S with respect to its
free variable as determined by FINDSYM.
 DIFF(S, 'v') or DIFF(S, sym('v')) differentiates S with respect to v.
 DIFF(S, n), for a positive integer n, differentiates S n times.
 DIFF(S, 'v', n) and DIFF(S, n, 'v') are also acceptable.

   Examples;
     x = sym('x');
     t = sym('t');
     diff(sin(x^2)) is 2*cos(x^2)*x
```

```
        diff(t^6, 6) is 720.

    See also INT, JACOBIAN, FINDSYM.
```

由于 help 命令的查找范围只限于 MatLab 的内部函数和命令，因此也就限制了它的使用范围。当查找的命令不是 MatLab 的函数，而是仅由它使用的属于 MAPLE V 的函数时，help 命令是无法找出结果的。这时就要使用 MAPLE V 的帮助命令 mhelp。它的使用格式与 help 命令完全相同，但查得的结果则是 MAPLE V 中的函数命令的帮助信息。

例 37

试查找在 MAPLE V 中的 diff 命令的帮助信息

```
mhelp diff
```

结果为：

```
Function: diff or Diff - Differentiation or Partial Differentiation
Calling Sequence:
   diff(a, x1, x2, ..., xn)
   Diff(a, x1, x2, ..., xn)
   diff(a, [x1, x2, ..., xn])
   Diff(a, [x1, x2, ..., xn])

Parameters:
   a              - an algebraic expression
   x1, x2, ..., xn - names
Description:
   - diff computes the partial derivative of the expression a with respect
to x1, x2, ..., xn, respectively. The most frequent use is diff(f(x),
x), which computes the derivative of the function f(x) with respect to x.
   - Note that where n is greater than 1, the call to diff is the same
as diff called recursively. Thus diff(f(x), x, y); is equivalent to the
call diff(diff (f(x), x), y);
   - diff has a user interface that will call the user's own differentiation
functions. If the procedure `diff/f` is defined then the function call
diff(f(x, y, z), y) will invoke `diff/f`(x, y, z, y) to compute the derivative.
    See example below.

Examples:
> diff(sin(x), x);
                                cos(x)
> diff(f(x, y), x, y);
                                2
                                d
                                ----- f(x, y)
                                dy dx
> diff(f(x, y), x, y) - diff(f(x, y), y, x);
                                0
```

```
# These two forms are equivalent:
See Also: D, int, dsolve, $, implicitdiff
```

注：以上内容删去了大部分详细的说明，只保留了 MAPLE V 中帮助信息的大体结构。

3.13 习　　题

(1) 基本画图功能——函数二维图形的表达

分别用 plot，fplot，ezplot 命令绘出函数 $y=x^2\sin x$ 在[-4，4]区间的图像。

(2) 不同精度的运算

① 体会相同的运算在 3 种不同精度下的速度差别

② 分别用 1 位和 10 位有效数字进行如下运算：

a.pi+2.4971

b.黄金分割点

(3) 使用 maple 命令

(4) MAPLE V 中的特殊函数及其使用方法

(5) 如何获得符号运算函数及命令的帮助信息

结合第 9、10、12 节内容，学习使用下列命令：

- 特殊函数 Zeta
- 特殊函数 BesselI
- convert

第 4 章　MatLab 在高等代数中的应用(基本篇)

利用 MatLab 强大的矩阵处理功能，可以非常方便地解决高等代数中的一些看似非常复杂的问题，如线性方程组的求解等。在第 4 章中，主要介绍如何利用 MatLab 求得代数矩阵的基本参数，如何建立、生成和修改矩阵，如何求解线型方程组，如何利用 MatLab 语言自己编写程序求解三维向量运算。

4.1　代数矩阵的基本运算与性质

代数矩阵的基本运算包括加、减、乘、除和乘方等运算，由于 MatLab 的所有运算是基于矩阵的，因而这些运算在 MatLab 中是最基本的计算，可以非常容易地实现。代数矩阵还有许多性质，如行列式的值、矩阵的迹和逆、条件数、秩等。绝大多数这些参数手工计算都是非常繁琐的，但在 MatLab 中有直接的函数可以利用，这正充分体现了 MatLab 强大的计算功能。下面将分别介绍这些运算。

4.1.1　矩阵的四则运算

在线性代数中，矩阵的加、减、乘运算是有明确定义的，在第 2 章中已经作了具体介绍，而矩阵除法确实没有。但在 MatLab 中为了便于计算，除法是有定义的，并且有左除和右除之分，这些在第 2 章中都已经作了介绍，下面举例来说明其应用。

例 1　设 A 和 B 两个工厂对 3 个产品的两个不同型号的产量分配如下，试求出 A 和 B 两个工厂对各个不同产品的不同型号的产量之和

```
   产品 1    产品 2    产品 3
  A=[8         0         3            型号 1
     3         6         0    ]       型号 2
       产品 1    产品 2    产品 3
  B=[2         5         2            型号 1
     1         0         7    ]       型号 2
```

计算 A+B 可以得到产品总产量的矩阵为：

```
A+B=[10        5           5
     4         6           7 ]
```

例 2　设 A，B 2 家机床厂生产 3 种机床，其月产量分别为

```
        机床 1    机床 2    机床 3
a=[   6        2        8          A厂
      0        6        10   ]     B厂
```

如果这 3 种机床的利润(单位：万元/台)分别为：

```
b=[1.2       机床 1
 2.3         机床 2
 0.8]        机床 3
```

求 A 和 B 这两个厂的月利润，其结果就是 a 和 b 两个矩阵的乘积，即：

```
a×b=[18.2000
  21.8000]
```

A 厂的月利润为 18.2 万元，B 厂的月利润为 21.8 万元。

以上两个例子非常简单，用手工算法计算还是可以的，然而在大宗的数据计算中，特别是矩阵的乘法，用手工计算是非常繁琐的，也容易出错。而利用 MatLab 强大的矩阵计算功能，如同在演算纸上进行计算一样快捷方便。

注意：在矩阵乘法中，两个非零矩阵的乘积可能为零；也就是说由 A*B=0 并不能推出 A=0 或 B=0。如下面的式子：

```
A=[ 1  0  0
    1  0  0
    0  0  0]
B=[ 0  0  0
    0  0  0
    1  1  1]
```

计算可得 A*B=0，但 A 和 B 都不是零矩阵。

4.1.2 矩阵的转置

矩阵的转置计算在 MatLab 中非常简单，就是运算符“'”，此运算符的运算级别比加减乘除等运算要高。

```
a =[  6    2    8;   0    6    10]
a'=[     6    0
         2    6
         8    10]
```

如果一个矩阵与其转置矩阵相等，则称其为对称矩阵；如果一个矩阵与其转置矩阵的和为零矩阵，则称其为反对称矩阵。在 MatLab 中判断对称和反对称的方法非常简单，如果 isequal(A，A')返回 1，则矩阵 A 为对称矩阵；如果 isequal(A，-A')返回 1，则矩阵 A 为反对称矩阵。

例 3

```
D =[ 1    2    3
     2    2    2
```

```
          3     2     4]
   isequal(D,D')=1
```

4.1.3 方阵的行列式

对于方阵的行列式，手工计算是非常繁琐的，尤其是对于高阶方阵。而在 MatLab 中，利用 det(A)函数就可以非常简单地计算矩阵的行列式的值。由线性代数的知识可以知道，如果方阵 A 的元素为整数，则计算结果也为整数。

例 4 计算下面方阵的行列式的值

```
C=[ 1 0 2 -5
-1 2 1 3
2 -1 0 1
1 3 4 2]
det(C)=0
```

说明：利用 det(X)==0 来检验方阵是否为奇异矩阵，并不是任何时候都有效的，在某些情况下就会判断错误。而利用 abs(det(X))<=tol 来判断矩阵的奇异性也并不可靠，因为误差 tol 是很难确定的，通常利用条件数 cond(X)来判断奇异或接近奇异的矩阵比较合理。在 MatLab 中，计算矩阵行列式的值时是采用高斯消去法来进行计算的，其计算的方法大体与下面的计算过程相同：

```
[l, u]=lu(A)              %对方阵 A 进行 LU 分解
a=det(l)                  %值为+1 或-1
a*prod(diag(u))           %此结果即为 A 的行列式的值
```

例 5

```
A =[
    11    10     4
    12    11   -13
    14    13   -66]
det(A) = -19
[l,u]=lu(A);
det(l)*prod(diag(u))
ans =-19.0000
```

还可以利用 MatLab 强大的计算功能来推算一些类似不定维方阵的行列式的值，这对实际问题用处不是很大，如果灵活运用 MatLab，可以解决一些问题。

例 6 求下面方阵的行列式的值

```
[   1   2   2   …   2
    2   2   2   …   2
    2   2   3   …   2
    ⋮   ⋮   ⋮   ⋮   ⋮
    2   2   2   …   n   ]
```

编名为 dt.m 的 M 文件，内容如下：

```
function y=dt(n)
c=diag(1:n);
c(c==0)=2;
y=det(c);
```

然后分别利用此函数求得当 n=1～8 时的行列式的值：

```
dt(1) = 1        dt(2) = -2       dt(3) = -2       dt(4) = -4
dt(5) =-12  dt(6) =-48  dt(7)=-240  dt(8) =-1440
```

其中的规律是，从 n=2 开始，前后两个结果的比的倒数是以自然数规律变化的，这与阶乘非常相似。最后结果为 dt(n)=－2×(n-2)！(n>1)。

4.1.4　矩阵的逆和伪逆

对于方阵 A，如果为非奇异方阵，则存在逆矩阵 inv(A)，使得 A*inv(A)=I。手工计算矩阵的逆是非常繁琐的，而利用函数 inv(A)则会非常方便地求出方阵的逆。函数 inv(A)在求方阵的逆时采用高斯消去法。如果 A 不是方阵或 A 为奇异或接近奇异的方阵，函数会给出警告信息，计算结果将都为 Inf。在不是采用 IEEE 算法的机器上，上述情况将会出错。当函数找到方阵的逆，但认为计算结果并不可靠时，将会出现以下的警告信息：

```
Warning: Matrix is close to singular or badly scaled.
   Results may be inaccurate. RCOND = xxx
```

说明方阵可能是奇异的或病态的，此函数的算法并不能保证计算结果的正确性。

例 7

```
A =
   11    10     4
   12    11   -13
   14    13   -66
inv(A) =
    29.3158  -37.4737    9.1579
    -32.1053   41.1579  -10.0526
        -0.1053    0.1579   -0.0526
c =
    0     0     0
    0     1     0
    0     0     0
inv(c)
Warning: Matrix is singular to working precision.
ans =
    Inf   Inf   Inf
    Inf   Inf   Inf
        Inf   Inf   Inf
```

对于奇异方阵或非方阵的矩阵，并不存在逆矩阵，但可以用函数 pinv(A)求其伪逆。其基本语法为 X=pinv(A)，X=pinv(A，tol)。函数返回一个与矩阵 A'具有相同大小的矩阵 X，并且满足：AXA=A，XAX=X，且 AX 和 XA 都是 Hermintian 矩阵。计算方法是基于奇异

```
function y=dt(n)
c=diag(1:n);
c(c==0)=2;
y=det(c);
```

然后分别利用此函数求得当 n=1～8 时的行列式的值：

```
dt(1) = 1        dt(2) = -2       dt(3) = -2       dt(4) = -4
dt(5) =-12  dt(6) =-48  dt(7)=-240  dt(8) =-1440
```

其中的规律是，从 n=2 开始，前后两个结果的比的倒数是以自然数规律变化的，这与阶乘非常相似。最后结果为 dt(n)=－2×(n-2)！(n>1)。

4.1.4　矩阵的逆和伪逆

对于方阵 A，如果为非奇异方阵，则存在逆矩阵 inv(A)，使得 A*inv(A)=I。手工计算矩阵的逆是非常繁琐的，而利用函数 inv(A)则会非常方便地求出方阵的逆。函数 inv(A)在求方阵的逆时采用高斯消去法。如果 A 不是方阵或 A 为奇异或接近奇异的方阵，函数会给出警告信息，计算结果将都为 Inf。在不是采用 IEEE 算法的机器上，上述情况将会出错。当函数找到方阵的逆，但认为计算结果并不可靠时，将会出现以下的警告信息：

```
Warning: Matrix is close to singular or badly scaled.
   Results may be inaccurate. RCOND = xxx
```

说明方阵可能是奇异的或病态的，此函数的算法并不能保证计算结果的正确性。

例 7

```
A =
   11   10    4
   12   11  -13
   14   13  -66
inv(A) =
   29.3158  -37.4737    9.1579
   -32.1053   41.1579  -10.0526
       -0.1053    0.1579   -0.0526
c =
    0     0     0
    0     1     0
    0     0     0
inv(c)
Warning: Matrix is singular to working precision.
ans =
    Inf   Inf   Inf
    Inf   Inf   Inf
        Inf   Inf   Inf
```

对于奇异方阵或非方阵的矩阵，并不存在逆矩阵，但可以用函数 pinv(A)求其伪逆。其基本语法为 X=pinv(A)，X=pinv(A，tol)。函数返回一个与矩阵 A'具有相同大小的矩阵 X，并且满足：AXA=A，XAX=X，且 AX 和 XA 都是 Hermintian 矩阵。计算方法是基于奇异

```
Warning: Rank deficient, rank = 2  tol =  1.2993e-014.
 f2 =
   -0.0909
    0.4545
         0
F*f1 =
    0.7273
    1.9091
    3.0000
    4.0909
F*f2 =
    0.7273
    1.9091
    3.0000
    4.0909
```

以上两个解不同的原因将在“线性方程组的解”一节中进行讨论。

4.1.5 矩阵的迹

矩阵的迹就是矩阵对角线元素的和，它等于矩阵的特征值之和。函数 trace(A)返回矩阵 A 的迹，它等价于 sum(diag(A))，这也是函数 trace()的定义式。

例 10

```
trace(E)
ans = 16
```

4.1.6 矩阵和向量的范数

线性代数中，在对线性方程组进行直接求解时，由于实际的观测和测量误差以及计算过程中的舍入误差等的影响，所求得的数值解与精确解之间存在着一定的差异，为了了解所得解的精确程度，必须对解的误差进行估计，这就要用到范数的概念。矩阵范数是矩阵元素的数量级大小的一种度量。在 MatLab 中，利用函数 norm()来求矩阵和向量的范数，其基本用法有：

n=norm(A) 返回矩阵 A 的最大奇异值，即 max(svd(x))，所得的是矩阵的 2-范数，也称普范数。

n=norm(A，1) 返回矩阵 A 的最大列之和，即 max(sum(abs((A))))，所得的是矩阵的 1-范数，也称作列范数。

n=norm(A，2) 与 n=norm(A)相同。

n=norm(A，inf) 返回矩阵 A 的最大行之和，即 max(sum(abs(A'))))，所得的是矩阵的无穷大范数，也称作行范数。

n=norm(A，'inf') 与 n=norm(A,inf)相同。

n=norm(A，'fro') 返回 sqrt(sum(diag(A'*A)))，即矩阵的 F-范数。

对于向量有：

n=norm(v，p) 返回 sum(abs(v).^p)^(1/p)。

n=norm(v)　得到 norm(v,2)。

n=norm(v，inf)　得到 max(abs(v))。

n=norm(v，－inf)　得到 min(abs(v))。

例 11

```
norm(E) = 24.0210
norm(E,1) =30
norm(E,2) =24.0210
norm(E,inf) =33
norm(E,'fro') =25.5539
```

4.1.7 矩阵的条件数

矩阵的条件数也是刻画线性方程组的解的精确性的一个非常重要的参数，它反映了线性方程组的解系随数据中的误差变化的敏感程度，是反映矩阵的逆以及所组成的线性方程组的解的精度的指标。矩阵的条件数越接近于 1，表明计算结果越精确。

C=cond(X)　是矩阵的最大奇异值与最小奇异值之比，这比值是 2-范数的矩阵条件数。

C=cond(X，1)　返回矩阵的 1-范数条件数。

C=cond(X，2)　等同于 cond(X)。

C=cond(X，'fro')　返回矩阵的 F-范数条件数。

C=cond(X，inf)　返回矩阵的无穷范数条件数。

在求解方程 AX=b 中，矩阵 A 的条件数越大，所求得的解受 A 和 b 的数据的误差的影响越大。

例 12

```
cond(A)
ans = 5.0961e+003
c=[1        1    1]'
A\c
ans =                         %方程组AX=c有精确解[1      -1   0]'
    1.0000
   -1.0000
    0.0000
```

现将 A 中的元素改变 0.01，则得：

```
a=[ 10.99   9.99          3.9
12.01   10.99             -13.01
14.01   13.01             -66.01];
a\c
ans =                         %此解与其原来的解已经完全不同
    4.0623
   -4.3637
   -0.0130
C =[  8     1     6
      3     5     7
```

```
    4     9     2]
cond(C) = 4.3301              %相比之下更接近 1
C\c
ans =                         %方程组 CX=c 的解
    0.0667
    0.0667
    0.0667
D=[8.01 1.01 5.99             %将矩阵 C 中的元素改变 0.01
   2.99 5.01 6.99
   4.01 9.01 2.01];
D\c
ans =                         %矩阵 DX=c 的解，与 CX=c 的解相差不多
    0.0663
    0.0665
    0.0670
```

4.1.8　矩阵的秩

矩阵的秩用函数 rank()来求得，函数返回矩阵的行向量或列向量的不相关个数。

rank(A)　返回矩阵 A 的奇异值中比误差 tol 大的奇异值的个数，tol 的默认值为 max(size(A))*norm(A)*eps。rank(A，tol)将指定误差 tol。

Matlab 中，计算矩阵的秩的算法是以奇异值分解为基础的，用奇异值分解的方法来求解矩阵的秩会非常耗时，但却是最有效的方法，可以用下面的方法来求矩阵的秩：

```
s = svd(A);
tol = max(size(A))*s(1)*eps;
r = sum(s > tol);
```

例 13

```
B =[ 0     1     2
     1     1    -1
     2     4     2]
rank(B) = 2
s=svd(B)
s =  5.2347
     2.1444
     0.0000
```

从矩阵 B 的奇异值可以看出，矩阵 B 的秩为 2。

4.2　矩阵的建立和修改

在 MatLab 中，最简单也是最直观的建立矩阵的方式，就是在 MatLab 的工作空间中直接输入矩阵，但在大多数情况下，这并不是最好的方法。本节将介绍用不同的方法建立矩阵、利用各种函数修改已经存在的矩阵、以最快捷的方式建立和生成需要的矩阵。

4.2.1 由文件生成和保存矩阵

在 MatLab 中输入矩阵，除在 MatLab 的工作空间中直接输入数据外，还可以利用各种文件来创建和保存矩阵。总结起来一般有 3 种方式：由 M 文件创建，由文本文件创建，由 MET 文件保存。在此举例说明如下。

例 14 分别建立名为 mfile.m 的 M 文件，内容为：

```
A=[0.9501    0.4860    0.4565
   0.2311    0.8913    0.0185
   0.6068    0.7621    0.8214];
```

名为 txtfile 的 TXT 文件，内容为：

```
8    1    6
3    5    7
4    9    2
```

然后在 MatLab 中执行下列命令：

```
clear                    %清除当前工作空间中的变量
mfile                    %执行 M 文件
who                      %查看工作空间中的变量
Your variables are:
A                        %变量 A 存在
load txtfile.txt         %装入 txtfile.txt 文件
who                      %查看变量
Your variables are:
A          txtfile       %共有 A 和 txtfile 两个变量
save mexfile             %保存工作空间变量到文件 mexfile
clear
who                      %无现实结果，即当前工作空间中无变量
load mexfile             %装载工作空间文件
who
Your variables are:
A               txtfile
txtfile =                %查看变量的内容
     8     1     6
     3     5     7
     4     9     2
```

从例 14 中可以看出 3 种文件建立和保存矩阵数据的作用。

4.2.2 由函数生成矩阵

对于经常用到的一些特殊的矩阵，如单位阵、全 0 阵、全 1 阵、随机矩阵魔方阵、对角阵等，MatLab 都提供了相应的函数来快速地生成这些矩阵。可以灵活运用这些函数和矩阵修改的一些操作，方便地生成一些特殊格式的矩阵，如，利用对角阵和矩阵的左右翻转函数，可以生成矩阵元素在从左下角到右上角的对角线上的特殊矩阵。

1. 生成单位阵

eye(n)　　生成 n 阶单位阵。
eye(m，n)　　生成 m×n 阶单位阵，对角线上的元素为 1，其余的元素为 0。
eye([m，n])　　同 eye(m，n)。
eye(size(A))　　生成与矩阵 A 具有相同大小的单位阵。
此函数不能生成高阶单位阵，否则会出错。
例 15

```
eye(3)
ans =
     1     0     0
     0     1     0
     0     0     1
A=rand(2,4);
A=eye(size(A))
A =
     1     0     0     0
     0     1     0     0
```

2. 生成全 1 矩阵

ones(n)　　生成 n×n 的全 1 阵。
ones(m，n)　　生成 m×n 的全 1 阵。
ones([m，n])　　生成 m×n 的全 1 阵。
ones(n1，n2，n3，…)　　生成 n1×n2×n3×…的全 1 阵。
Ones([n1 n2 n3 …])　　生成 n1×n2×n3×…的全 1 阵。
Ones(size(A))　　生成与矩阵 A 具有相同大小的全 1 阵。
例 16

```
C=ones(2,4)
C =
     1     1     1     1
     1     1     1     1
```

3. 生成全 0 阵

zeros(n)　　生成 n×n 的全 0 阵。
zeros (m，n)　　生成 m×n 的全 0 阵。
zeros ([m，n])　　生成 m×n 的全 0 阵。
zeros (n1，n2，n3，…)　　生成 n1×n2×n3×…的全 0 阵。
zeros ([n1 n2 n3 …])　　生成 n1×n2×n3×…的全 0 阵。
zeros (size(A))　　生成与矩阵 A 具有相同大小的全 0 阵。

在 MatLab 中，没有变量生成的必要，所有变量的存储空间都是在给变量赋值时自动分配的，然而频繁分配变量空间会大大降低语句的执行速度，因而应该尽量避免不必要的

频繁的变量空间的分配。通常先用 zeros()函数给变量分配好足够大小的空间，再对变量进行赋值。

例 17　依次执行下面的语句：

```
tic                          %开始计时
for i=1:10000
   c(i)=i;                   %每次都要重新分配空间
end
toc                          %读取计时时间
tic                          %开始计时
d=zeros(1,10000);            %预先分配空间
for i=1:10000
   d(i)=i;                   %每次赋值，不必分配空间
end
toc                          %读取计时时间
```

结果如下：

```
elapsed_time =               %第 1 次读计时数
    9.5100
elapsed_time =               %第 2 次读计时数
    0.1600
```

从例 17 可以看出，第 2 次赋值所用的时间要比第 1 次少得多(以上是在主频为 200MHZ 的机器上运行的结果)。

4. 生成随机阵

随机阵在某些场合中还是比较有用的，下面介绍一下。

(1) 在[0，1]区间均匀分布的随机阵：

- rand(n)　　生成 n×n 的随机阵。
- rand (m，n)　　生成 m×n 的随机阵。
- rand ([m，n])　　生成 m×n 的随机阵。
- rand (n1，n2，n3，…)　　生成 n1×n2×n3×…的随机阵。
- rand([n1 n2 n3 …])　　生成 n1×n2×n3×…的随机阵。
- rand(size(A))　　生成与矩阵 A 具有相同大小的随机阵。
- rand　　返回在[0，1]区间上的随机数。
- rand('state')　　返回一个含有 35 个元素的列向量，反映当前随机发生器的状态。要改变其状态，可以用以下命令：
- rand('state'，s)　　将发生器的状态置为 s，s 可以是表达式。
- rand('state'，0)　　将发生器置为初始状态。
- rand('state'，n)　　将发生器置为第 n 个状态。
- rand('state'，sum(100*clock))　将发生器状态每次都改变。

说明：高版本的 MatLab 的随机发生器采用的是多种子发生器，理论上可以产生 2^{1492} 个不重复的数分布在区间[2^{-53}，$1-2^{-53}$]上。而低版本的发生器采用单种子发生器，用

rand('seed', s)则使用单种子发生器，s 可以指定发生器的种子；rand('seed')可以查看当前单种子发生器的种子。rand('state', s)将切换到多种子发生器。

例 18

```
rand(2,4)
ans =
    0.5162    0.1837    0.4272    0.8215
0.2252    0.2163    0.9706    0.3693
rand('seed')
ans =
  1.6915e+009
```

(2) 服从 N(0, 1)正态分布的随机阵：

函数 randn()来生成正态随机阵，其用法与 rand()的用法完全相同，在此不再赘述。

例 19

```
randn(2,4)
ans =
   -0.4326    0.1253   -1.1465    1.1892
   -1.6656    0.2877    1.1909   -0.0376
randn('seed')
ans =
   931316785
randn('state',1)
randn('state')
ans =
   362436069
           1
```

5. 生成魔方阵

魔方阵在第 1 章已经介绍过，其各行、各列及 2 对角线的元素和都相同。函数 magic(n)生成 n 阶魔方阵。

如 3 阶魔方阵：

```
8    1    6
3    5    7
4    9    2
```

可以验证其各行和各列的和均相等。

6. 生成对角阵

diag(v, k)　此函数返回矩阵的第 k 列对角线由向量 v 组成，k 可以是正数、负数或 0，如果 k 为 0，则由向量 v 构成矩阵的主对角线。diag(v)相当于 diag(v, 0)。

例 20

```
v =[ 1    1    1]
diag(v,-1)
```

```
ans =
     0     0     0     0
     1     0     0     0
     0     1     0     0
     0     0     1     0
```

4.2.3 矩阵的标识

矩阵元素的标识在对矩阵的单个或多个元素进行引用、赋值和修改中起着非常重要的作用，熟练掌握各种标识方式可以方便灵活地对矩阵进行修改和援引。矩阵的标识主要有元素标识方式、向量标识方式和0－1标识方式，与find()函数连用还有矩阵标识方式。

(1) 元素标识方式 A(i，j)

整数i和j分别标示元素在矩阵A中的行数和列数。

例 21

```
A =[ 8     1     6;  3     5     7;   4     9     2]
A(2,3) = 7
```

(2) 向量标识方式 A(vr，vc)

vr和vc分别为含有矩阵A的行号和列号的单调向量，vr和vc中如一个是"："，则表示取全部行(vr为"：")或全部列(vc为"：")。向量中的元素必须合法，即不会超出矩阵的维数。如A(1：3，[3 5 7])表示取矩阵A的第1，2，3行和第3，5，7列位置上的元素；A(：，[2　4])表示取矩阵A的第2和第4列的所有元素。

例 22

```
A(1,[1 3]) = 8     6
A(:,[1 3])
ans =
     8     6
     3     7
     4     2
```

(3) 0－1向量标识方式　A(v1，v2)，A(：，v2)，A(v1，：)

v1和v2是由元素0和1组成的长度分别等于矩阵A的行维和列维的向量，其中的元素如果取1则表示取此相应位置上的行或列，若为0则不取。v1和v2都应该为逻辑向量，否则会出错。

例 23

```
l1=[0 1 0]
l2=[1 1 0]
l1=l1==1
l2=l2==1
A(l1,l2)
ans =
     3     5
A(l1,:)
ans = 3     5     7
```

(4) 矩阵表示方式 A(find(B))

矩阵B是与A具有相同大小的矩阵，如果B的元素为非0值，则取当前位置的元素，否则不取。

例 24

```
D=A;
D(find(eye(3)))=0
D =
     0     1     6
     3     0     7
     4     9     0
D(find(~D))=inf
D =
     Inf     1     6
       3   Inf     7
       4     9   Inf
```

4.2.4 矩阵的修改和抽取

矩阵的修改包括对矩阵进行扩充、矩阵元素的部分删除、矩阵元素的抽取以及矩阵的结构变化等操作。这些操作在矩阵的构造和生成中具有非常重要的作用，灵活运用这些操作可以非常方便地生成所需的矩阵，对于矩阵元素较多的矩阵，避免了冗长的矩阵输入，节省了时间，并且不易出错。

1. 矩阵的扩充

矩阵的扩充一般用“[ ]”，用小的矩阵来生成大的矩阵，在构造过程中要保证生成矩阵的各行和各列的元素数相同，否则会出错。

例 25

```
A =
     1     1     1     1
     1     1     1     1
B=zeros(1,2)
B =[0     0]
C=[inf inf]
D=[ A
B C]
D =
     1     1     1     1
     1     1     1     1
     0     0   Inf   Inf
D=[a                        %此构造不能构成矩阵
B]
??? All rows in the bracketed expression must have the same
number of columns.
```

对超出矩阵的维数的元素直接进行赋值，也可以对矩阵进行扩展，其余未赋值的元素用 0 补齐。

例 26

```
a(1:3,1:3)=eye(3)
a =
     1     0     0     1
     0     1     0     1
     0     0     1     0
```

2. 矩阵的部分删除

在 MatLab 中，可以将矩阵的某行和某列赋值为空值而直接删除此行或列。删除行或列的部分元素则会出错。

例 27

```
a(:,2)=[]
a =
     1     0     1
     0     0     1
     0     1     0
a(2,3)=[]
???  Indexed empty matrix assignment is not allowed.
```

3. 矩阵的修改

利用下标直接进行修改，还可以利用"："进行提取和修改。A(:)表示将 A 的所有元素展成一列，A(:)如果在赋值语句的左边，则表示利用= =符号右边的数据重新组成矩阵 A，A 的维数不变。

例 28

```
B =
    0.9501    0.6068    0.8913
    0.2311    0.4860    0.7621
A=B(:)
A =
    0.4565
    0.0185
    0.8214
    0.4447
    0.6154
    0.7919
A=zeros(3,2);
A(:)=B
A =
    0.4565    0.4447
    0.0185    0.6154
```

```
      0.8214    0.7919
```

在高版本的 MatLab 中，利用命令 edit vname 可以调用 MatLab 的窗口界面直观地对变量进行结构和数据的修改，对其中的数据进行删除和扩充。

4. 元素的抽取

diag(A，k)　抽取矩阵 A 的第 k 对角线的元素组成列向量，k=0，将提取主对角线的元素。

diag(A)　抽取矩阵 A 的主对角线元素组成列向量。

例 29

```
diag(a,-1)
ans =
     0
          1
```

tril(A，k)　tril(A)提取矩阵的下三角部分。

例 30

```
b=ones(3)
tril(b,-1)
ans =
     0      0      0
     1      0      0
     1      1      0
```

triu(A，k)　triu(A)提取矩阵的上三角部分。

例 31

```
triu(b,1)
ans = 0      1      1
     0      0      1
     0      0      0
```

5. 矩阵结构的改变

(1) reshape(A，n1，n2，n3，…)和 reshape(A，[n1，n2，n3，…])　返回以矩阵 A 的元素作为元素的 n1×n2×n3×…维矩阵，A 的元素个数等于 prod(n1，n2，n3，…)。

例 32

```
reshape(b,1,9)
ans =
     1     1     1     1     1     1     1     1     1
```

(2) rot90(A)　将矩阵 A 逆时针旋转 90°。

rot90(A，k)　将矩阵 A 逆时针旋转 k×90°，k 为整数。

例 33

```
A =
```

```
    0.9218    0.1763    0.9355
0.7382    0.4057    0.9169
rot90(A)
ans =
    0.9355    0.9169
    0.1763    0.4057
    0.9218    0.7382
```

(3) fliplr(A)　将矩阵 A 左右翻转。

例 34

```
fliplr(A)
ans =
    0.9355    0.1763    0.9218
          0.9169    0.4057    0.7382
```

(4) flipud(A)　将矩阵 A 上下翻转。

例 35

```
flipud(A)
ans =
    0.7382    0.4057    0.9169
          0.9218    0.1763    0.9355
```

4.3　线性方程组求解

在高等代数中，一个非常重要的内容就是对线型方程组的求解。在实际的工程应用、进行实验数据的分析、进行理论研究等很多情况下，很多问题都可以归结为高等代数的线型方程组的求解，因而线性方程组的求解问题的应用是非常广泛的。在这一节中，将讲述 MatLab 对恰定方程组、超定方程组、欠定方程组、病态方程组等的求解。

4.3.1　解恰定方程组

对于方程 Ax=b，如果 A 为方阵，则此方程称为恰定方程。解恰定方程可以用方阵的逆来求，即：x=inv(A)b。

例 36

```
A =[
    1    -2    -3    -4
    2     1    -1     1
   -1     0    -1     2
    3    -3     4    -5];
x=inv(A)*b
x =
  -1.5000
   2.7500
```

```
    0.0000
       -2.7500
```

如果 A 是奇异方阵，则计算结果为 Inf，且会给出警告信息。如果矩阵 A 为病态矩阵，也会给出警告信息。

此外，还可以用除法来解恰定方程，对于上面的方程，其解为 x=A\b。

上面两种方法的区别是，从算法上说，上面的两种计算方法都采用高斯消去法，但利用除法求解时，不是先对矩阵 A 求逆，而是直接利用高斯消去法进行计算。这样可以很好地保证计算的精度，而又会节省大量的计算时间。从例 37 中可以明显看出除法的优势。

例 37

```
A=rand(100)+1.e10;                    %生成 100×100 随机阵
x=ones(100,1);
b=A*x;                                %求方程右边的向量
tic                                   %开始计时
y=inv(A)*b                            %用求逆来解方程
toc                                   %读计时时间
err = norm(y-x)                       %结果与精确解的范一2 误差
res = norm(A*y-b)                     %方程的范一2 误差
tic
y=A\b                                 %用除法解方程
toc
err = norm(y-x)
res = norm(A*y-b)
elapsed_time =                        %求逆解方程所用时间
    0.1600
err =
    0.0436
res =
  5.3696e+009
elapsed_time =                        %除法解方程所用时间
    0.0500
err =
    0.0142
res =
    0.0036
```

说明：从执行的结果可以看出，除法解方程所用的时间大约只有求逆法的 1/3，其误差也远远小于求逆法，因此在求方程的解时应该尽量使用除法运算。

4.3.2 解超定方程

对于方程 Ax=b，A 为 n×m 矩阵，如果 A 列满秩，且 n>m，则方程是没有精确解的，即其精确解的空间为零。然而在实际工程应用中，求得其最小二乘解也是有意义的。方程的解除了可以用除法运算来求(x=A\b)外，还可以用广义逆来求：x=pinv(A)，所求的解并不

满足 Ax=b，x 只是最小二乘意义上的解。

例 38

```
C=[3 4 5
   6 1 2
   4 -5 7
   8 2 4]
c=[3  2  4  6]'
rank(C) =  3
x1=C\c
x1 =
    0.4149
    0.0448
    0.3737
C*x1-c
ans =
    0.2924
    1.2815
    0.0516
   -1.0966
```

可见 x1 并不是方程 C*x1=c 的精确解，用 x2=pinv(C)*c 所得的解与 x1 相同，用线性代数可以证明，列满秩方程 CX=c 的最小二乘解为 X= inv(C'*C)*C'*c，而 pinv(C)= inv(C'*C)*C'。如例 38 的结果可以这样计算：

```
inv(C'*C)*C'*c
ans = 0.4149
      0.0448
      0.3737
```

4.3.3 解欠定方程

欠定方程从理论上说是有无穷多个解的，如果利用求逆和矩阵除法来求解，只能得到其中的一个解。对于例子：

```
B =[ 1    -2     3
     0     1    -1
    -1     0    -1
     1    -3     4]
   b=[ 4    -3    -4     1]
   x=pinv(B)*b
x = 2.2549
    1.2157
    1.0392
 y=B\b
Warning: Rank deficient, rank = 2  tol =   4.6151e-015.
y =  0
    3.4706
```

```
        3.2941
```

x 和 y 都是方程的解，其中 x=pinv(B)*b 是所有解中范数最小的一个，norm(x) = 2.7645，而 norm(y) = 4.7850。y=B\b 是所有解中 0 最多的一个，也就是非零元素最多的一个。

4.3.4 解齐次线性方程组

对于方程 AX=0，MatLab 中有一个非常有用的函数 B=null(A)，返回矩阵 A 的零空间的标准正交基组成的矩阵 B，使得 B'×B=1，B 的列数等于矩阵 A 的零空间的维数。对于任意非齐次线性方程组 AX=b，用 4.3.2 节介绍的方法求出方程的一个特解 X0，再用函数 null(A)求出齐次线性方程 AX=0 的解空间 B，然后用 B×Y+X0 即可以将方程所有的解表示出来，其中 Y 为维数等于矩阵 A 的零空间维数的列向量。

例 39

```
A =[  1    -2     3    -4
      0     1    -1     1
     -1     0    -1     2
      1    -3     4    -5]
a=null(A)
a =  0.7888   -0.2110
    -0.4999   -0.2889
    -0.2110   -0.7888
     0.2889   -0.4999

a'*a = 1.0000    0.0000
       0.0000    1.0000
```

下面给出另一个求矩阵 A 的零空间的程序，所求的解将含有最多的零元素个数。此函数只限于求方阵的零空间。

```
function C=solution(A)                          %A 应为方阵
[l,u]=lu(A);
M1=[];M2=[];e=[];f=[];c=[];
r=rank(A);
rf=size(A);
n=rf(1,2);
k=1;
for i=1:r
  while k<=n
    if u(i,k)~=0
      e=[e k];
      M1=[M1 u(1:r,k)];
      k=k+1;
      break;
    else
      f=[f k];
      M2=[M2 -u(1:r,k)];
    end
```

```
      k=k+1;
    end
  end
  M2=[M2 -u(1:r,k:n)];
  f=[f k:n];
  for i=1:(n-r)
    y=zeros((n-r),1);
    y(i)=1;
    x=M1\(M2*y);
    c=zeros(n,1);
    for j=1:r
      c(e(j))=x(j);
    end
    for j=1:(n-r)
      c(f(j))=y(j);
    end
    C=[C c];
  end
```

例 40 利用 solution()函数求矩阵 A 的零空间

```
e=solution(A)
e =
    -1     2
     1    -1
     1     0
     0     1
A*e
ans =
     0     0
     0     0
     0     0
     0     0
```

4.3.5 解病态方程组

在用 4.3 节的方法求解方程时，基本上都是用高斯消去法。用这种方法在解某些方程时虽然从理论上可以得到精确解，但是由于计算的误差，并不能得到很好的结果，还可能出现很大的偏差。下面几种情况下，方程可能出现病态：

- 用消元法解方程时出现了小主元。
- A 的行列式相对很小，或某些行或列近似线性相关。
- A 的元素数量级相差很大。
- A 的最大特征值和最小特征值的比值绝对值很大。

由于 MatLab 的计算的精度很高，在某些计算程序中可能出现病态的方程在 MatLab 中计算也会得到很好的结果，但不是所有的方程都这样。希望注意此类方程，要检验其解的正确程度，看是否出现病态。

4.3.6　解非负最小二乘

在某些情况下，所得线性方程组的解出现负数是没有意义的，虽然方程组可以得到精确解，但却不能取带有负数元素的解，在这种情况下，其非负最小二乘解要比方程的精确解更有用。在 MatLab 中，通常用 nnls()函数来求方程组的精确解。

nnls(A，b)　返回方程 Ax=b 的最小二乘解，方程的求解过程被限制在 x>=0 的条件下。

nnls(A，b，tol)　指定误差 tol 来求解，tol 的默认值为 max(size(A))*norm(A,1)*eps，矩阵的一1 范数越大，求解的误差越大。

[x，w]=nnls(A，b)　同时返回一个双向量 w，当 x(i)=0 时，w(i)<=0；当 x(i)>0 时，w(i)=0。

[x，w]=nnls(A，b，tol)　指定求解误差 tol。

例 41

```
A =[3.4336   -0.5238    0.6710   -0.1527
   -0.5238    3.2833   -0.7302   -0.2689
    0.6710   -0.7302    4.0261   -0.0984]
b =[ -1.0000    1.5000    2.5000   -2.0000]
   -0.1527   -0.2689    0.0184    2.7507
[x,w]=nnls(A,b)
x =      0                                  w = -3.2832
    0.7056                                         0
    0.7142                                    0.0000
         0                                   -4.7804
a=A\b
a = -0.3942
    0.5055
    0.7611
          -0.7046
A*a-b =                                     A*x-b =
  1.0e-015 *
         0                                   1.1097
    0.2220                                   0.2953
   -0.4441                                  -0.1397
   -0.4441                                   1.8234
```

4.4　向 量 运 算

向量实际上是单行或单列数据，是矩阵的一种特殊形式。单行数据称作行向量，单列数据称作列向量。因为向量是一种特殊的矩阵，因而它满足一切矩阵的运算，包括加减乘除等运算，同时也必须满足矩阵运算的一切条件，如加减运算必须为同维数的行向量或列向量。

4.4.1 向量的生成

在 MatLab 创建向量时，除了在命令窗口中直接输入外，还可以用 MatLab 的一些操作和命令来自动创建。向量的自动生成主要有以下 3 种方法：

1. 由冒号生成

X=m1：m2 生成行向量 x=[m1+1，m1+2，…，m1+k]并满足 m2-1<m1+k≤m2。M1 和 m2 不一定为整数，但 m1 应小于或等于 m2，否则会返回空向量。

X=m1：d：m2 生成行向量 x=[m1+d，m1+2d，…，m1+kd]并满足 m2-d<m1+kd≤m2。M1 和 m2 不一定为整数，如 d<0，则 m1≤m2；d>0 则 m1≥m2，否则会返回空向量。

例 42

```
b=1:0
b =  Empty matrix: 1-by-0
b=3.4:6.7
b = 3.4000    4.4000    5.4000    6.4000
b=2.6:-0.8:0
b =2.6000    1.8000    1.0000    0.2000
```

2. 由 linspace()生成线性等分向量

L=linespace(x，y)　生成 100 维的行向量，使满足 l(1)=x，l(100)=y。

L=linespace(x，y，n)　生成 n 维的行向量，使满足 l(1)=x，l(n)=y。如果 n=1，则返回 y，即 L=y。

例 43

```
l=linspace(1,10,10)
l =  1     2     3     4     5     6     7     8     9    10
```

3. 由 logspace()生成对数等分向量

L=logspace(a，b)　生成 10^a～10^b 的 50 个对数等分点组成的行向量。

L=logspace(a，b，n)　生成 10^a～10^b 的 n 个对数等分点组成的行向量。

L=logspace(a，pi)　生成 10^a～pi 的 50 个对数等分点组成的行向量。

L=loaspace(a，pi，n)　生成 10^a～pi 的 n 个对数等分点组成的行向量。

将对数等分向量取 10 的对数变成线性等分向量。

例 44

```
c=logspace(1,5,5)
c = 10         100        1000       10000       100000
d=log10(c)
d = 1.0000    2.0000    3.0000    4.0000    5.0000
c=logspace(4,pi,5)
c = 1.0e+004 *
```

```
    1.0000    0.1331    0.0177    0.0024    0.0003
d=log10(c)
d = 4.0000    3.1243    2.2486    1.3729    0.4971
```

此外，还可以提取矩阵的某行或某列组成向量。

4.4.2 三维向量的运算

由于在三维空间坐标系中，向量和点都可以用一个三维向量来表示，因而三维向量的运算在整个向量运算中的意义非常重大。在此节中着重讲述三维向量的空间运算。

1. 向量共线和共面的判断

向量的共线或共面的判断可以用向量构成的矩阵的秩来判断。举例如下：

```
function y=iscline(a,b)
n=length(a);
y=1;
if n~=length(b)
   y=0;
   return;
end
if rank([reshape(a,n,1) reshape(b,n,1)])~=1
   y=0;
   return;
end
```

上例中如果 a 和 b 共线，函数返回 1，否则返回 0。

例 45 判断 3 个向量是否共面，如果共面返回 1：

```
a =[ 1    2    3]
b =[ 2    4    6]'
iscline(a,b) = 1

function y=iscface(a,b,c)
n=length(a);
y=1;
if n~=length(b)&n~=length(c)
   y=0;
   return;
end
if rank([reshape(a,n,1) reshape(b,n,1) reshape(c,n,1)])==3
   y=0;
   return;
end
```

例 46

```
c = [1    5    3]
d =  [2    4    7]
```

```
iscface(a,b,d) = 1
iscface(a,c,d)= 0
```

2. 向量的长度

向量的长度是向量元素平方和的平方根，其函数形式为：

```
function y=radvec(v)
y=sqrt(sum(sum(v.^2)));
```

例 47 求向量 c 的长度

```
radvec(c)
ans = 5.9161
```

向量除以其长度可以化成单位向量：

```
uc=c/radvec(c)
uc = 0.1690   0.8452   0.5071
```

3. 向量的方向角

在空间直角坐标系中，向量与 3 个坐标轴 X，Y，Z 之间的夹角 a，b，c 称作向量的方向角。方向角余弦称作向量的方向余弦。求方向角和方向余弦可以用下列函数来求：

```
function y=dangle(v,flag)
y=zeros(3,1);
if nargin==1
   flag=0;
end
d=radvec(v);
if d==0
   error('d is a 0-vector');
end
y(1)=v(1)/d;
y(2)=v(2)/d;
y(3)=v(3)/d;
if flag~=0
   y(1)=acos(y(1));
   y(2)=acos(y(2));
   y(3)=acos(y(3));
end
```

例 48 求向量 c 的方向余弦

```
d=dangle(c,0)
d =  0.1690
    0.8452
         0.5071
```

4. 向量的点积

两个向量的点积是两个向量相应元素的乘积和。

```
function y=dotab(a,b)
y=0;
for i=1:length(a)
   y=y+a(i)*b(i);
end
```

例 49

```
dotab(d,d) = 1
```

5. 向量的夹角

```
function y=vtov(v1,v2,flag)
if nargin==2
   flag=0;
end
if flag==0
   y=dotab(v1,v2)/radvec(v1)/radvec(v2);
elseif flag==1
   y=acos(dotab(v1,v2)/radvec(v1)/radvec(v2));
elseif flag==2
   y=acos(dotab(v1,v2)/radvec(v1)/radvec(v2))*180/pi;
end
```

例 50

```
vtov(d,d,2) = -3.5098e-015                    %结果存在计算误差
```

6. 点与点之间的距离

```
function y=dist(v1,v2)
y=radvec(v1-v2);
```

例 51 求点[1 2 3]与点[5 2 4]之间的距离

```
dist([1 2 3],[5 2 4])
ans = 4.1231
```

7. 向量的正交投影

```
function [c,d]=orthprj(a,b)
k=dotab(a,b)/(radvec(b)^2);
c=k*b;
d=a-c;
```

例 52

```
[c,d]=orthprj([1 2 3],[5 2 4])
```

```
c = 2.3333    0.9333    1.8667
d = -1.3333    1.0667    1.1333
c+d = 1    2    3
dot(d,[5 2 4])
ans = 2.6645e-015                                    %应该为 0，存在计算误差
```

8. 向量的向量积

向量的向量积是一个非常重要的参数，在求与面垂直的向量以及平行四边形的面积时非常有用。

```
function y=cross_product(a,b)
a=reshape(a,3,1);
b=reshape(b,3,1);
c=eye(3);
y(1)=-det([b a c(:,1)]);
y(2)=-det([b a c(:,2)]);
y(3)=-det([b a c(:,3)]);
```

例 53 求与向量[2 -3 1]及[3 0 4]都垂直的单位向量

```
v=cross_product([2 -3 1],[3 0 4])
v=v/radvec(v)
v = -0.7589   -0.3162    0.5692
otab(v,[2 -3 1])
ans = -1.1102e-016                                  %应该为 0，存在计算误差
```

例 54 求以 p1[1 2 3]，p2[2 0 5]，p3[4 2 -1]为顶点的三角形的面积

```
v1=p1-p2
v1 = -1    2    -2
v2=p1-p3
v2 = -3    0    4
c=cross_product(v2,v1)
c = -8   -10    -6
area=0.5*radvec(c)
area = 7.0711
```

9. 向量的混合积

用向量的混合积可以求平行六面体的体积。

```
function y=mixed_product(a,b,c)
y=dotab(cross_product(a,b),c);
```

例 55 求 p1[0 0 2]，p2[3 0 5]，p3[1 1 0]，p4[4 1 2]组成的四面体的体积

```
v1=p1-p2
v1 =-3    0    -3
v2=p1-p3
v2 =-1   -1     2
```

```
v3=p1-p4
v3 =-4    -1     0
d=abs(mixed_product(v1,v2,v3))/6     %四面体的体积为平行六面体的 1/6
d = 0.5000
```

10. 点到平面的距离

对于方程为 Ax+By+Cz+D=0 的平面，用四维向量[A B C D]表示，点用三维向量来表示。则求点到平面的距离的函数可以写为：

```
function y=ptof(p,f)
d1=dotab(f,[p 1]);
d2=radvec(f(1:3));
if d2==0
   error('平面矩阵输入错误');
else
   y=abs(d1/d2);
end
```

例 56　求点[3 2 4]到平面 2x−6y+3z+1=0 的距离

```
p=[3 2 4];
f=[2 -6 3 1];
ptof(p,f)
ans = 1
```

11. 点到直线的距离

用直线上的一点和代表直线方向的向量来表示直线，即将直线(x−x0)/A=(y−y0)/B=(z−z0)/C 表示为点 p[x0　y0　z0]和向量 v[A　B　C]。求点 p 到直线 v(方向向量)，vp(直线上一点)的距离的函数可以写为：

```
function y=ptol(p,v,vp)
vs=vp-p;
d=radvec(v);
if abs(d)<eps
   error('input argument error for v');
end
y=abs(radvec(cross_product(vs,v))/d);
```

例 57　求点 p[3 0 2]到直线(x−2)/2=(y−1)/1=z/1 的距离

```
p=[3 0 2];
v=[2 1 1 ]
vp=[2 1 0]
ptol(p,v,vp)
ans = 2.1213
```

12. 异面直线的距离

程序如下：

```
function y=ltol(v1,p1,v2,p2)
if iscline(v1,v2)
  error('arguments input error');
end
pp=p1-p2;
d=radvec(mixed_product(v1,v2,pp))/radvec(cross_product(v1,v2));
y=abs(d);
```

例 58 求异面直线 x=y/2=z/3 及 x-1=y+1=z-2 之间的距离

```
ltol([1 2 3],[0 0 0],[1 1 1],[1 -1 2])
ans = 2.0412
```

13. 点关于直线的对称点及垂足

pd 为点到直线的垂足，pp 为点关于直线的对称点。

```
function [pd,pp]=porthl(p,vp,v)
p=reshape(p,1,3);
vp=reshape(vp,1,3);
v=reshape(v,1,3);
[c,d]=orthprj(p-vp,v);
pd=vp+c;
pp=p-2*d;
```

例 59 求点 p[3 2 1] 关于直线 x/1=y/0=z/0 的垂足和对称点。

```
[pd pp]=porthl([3 2 1 ],[0 0 0],[1 0 0])
pd =  3     0     0
pp =   3    -2    -1
```

4.5 习　　题

(1) 已知 A=[1 3 5
2 1 4
1 2 3]　　B=[1 2 -1
2 0 1
3 1 2]

求 AB，AB-BA，A'B。

(2) 求下列矩阵的秩、行列式的值、迹及范数和条件数

[1 0 2 -5
-1 2 1 3
2 -1 0 1
1 3 4 2]　　[1 2 0 0
-1 3 0 0
0 0 2 -1
0 0 5 4]

$$\begin{bmatrix} 0 & 0 & 1 & -1 & 2 \\ 0 & 0 & 2 & 0 & -3 \\ 0 & 0 & -1 & 4 & 0 \\ -1 & 2 & 4 & 0 & -1 \\ 3 & -2 & 1 & 5 & 1 \end{bmatrix}$$

(3) 求解下列方程组

① $\begin{cases} -x_1+2x_2=2 \\ 2x_1+x_2+x_3=3 \\ 4x_1+5x_2+7x_3=0 \\ z_1+x_2+5x_3=-5 \end{cases}$　② $\begin{cases} x_1+x_2+3x_3-x_4=-2 \\ x_2-x_3+x_4=1 \\ x_1+x_2+2x_3+2x_4=4 \\ x_1-x_2+x_3-x_4=0 \end{cases}$

③ $\begin{cases} x_1-2x_2+3x_3-4x_4=4 \\ x_2-x_3+x_4=-3 \\ -x_1-x_2+2x_4=-4 \end{cases}$　④ $\begin{cases} x_1-4x_2+2x_3=0 \\ 2x_2-x_3=0 \\ -x_1+2x_2-x_3=0 \end{cases}$

(4) 判断下列各组向量是否共线

① x=[1，2，3]，y=[-1，-2，-3]，z=[3，-6，9]

② x=[1，0，2]，y=[2，0，-4]，z=[3，0，6]

(5) 判断下列各组向量是否共面

① x=[4，0，2]，y=[6，-9，8]，z=[6，-3，3]

② x=[1，-1，2]，y=[2，4，5]，z=[3，9，8]

(6) 求点 P 到平面 K 的距离

① P[-2，-3，1]，K：2x-y+2z+1=0

② P[2，-1，1]，K：16x-10y+10z-3=0

(7) 求异面直线 K1 和 K2 间的距离

① K1：(x+7)/3=(y+3)/4=(z+3)/-2，K2：(x-11)/6=(y+5)/-4=(z-2)/-1

② K1：(x+5)/3=(y+5)/2=(z-1)/-2，K2：(x-9)/6=y/2=(z-2)/-1

(8) 求点 P[2，-4，5]关于直线(x-2)/3=(y+3)/7=(z-1)/-3 的对称点及在此直线上的垂足。

(9) 求以 P1[1，-1，3]，p2[5，-6，2]，p3[1，3，-1]为顶点的三角形的面积及 P1P2 边上的高。

(10) x=[1，0，-1]，y=[1，2，1]，z=[-1，0，2]，求 xy，yz，xyz。

第 5 章　MatLab 在高等代数中的应用(高级篇)

本章将介绍利用 MatLab 实现求解特征值和特征向量、矩阵的对角化、求解矩阵的 Jordan 标准形、矩阵的各种分解及一元多项式的各种操作，并通过几个实例介绍了利用 MatLab 求解实际问题的应用。这些都是高等代数中比较深入的部分，单靠利用 MatLab 的函数并不能完全解决。本章给出了许多扩展的函数，凡是 MatLab 的标准版中没有给出的函数，在介绍的过程中都会给出其源代码，便于参考和借鉴。

5.1　特征值与特征向量

特征值与特征向量是线性代数中非常重要的概念，在实际的工程应用和在求解数学问题中占有非常重要的地位。例如在工程技术中，如振动问题、稳定问题等等，从数量关系上常常归结为求矩阵的特征值与特征向量的问题。在数学中，求解微分方程组以及简化矩阵计算等都要用到特征理论。特征值与特征向量的问题在与振荡有关的各个学科中，如机械、土木、电子、化工、生态学、核物理、气体力学、弹性力学等领域都有着广泛的应用。

5.1.1　特征值与特征向量的求解

在 MatLab 中函数 eig()可以用来求解矩阵的特征值与特征向量。特征值 k 的特征向量 x 与矩阵 A 之间的关系满足 Ax=kx。而对于矩阵 A 和 B，其广义特征值 k 和广义特征向量 x 之间应该满足 Ax=kBx。函数 eig()的基本用法有：

- d=eig(A)　返回由矩阵 A 的特征值组成的列向量。
- [V，D]=eig(A)　返回特征值矩阵 D 和特征向量矩阵 V。特征值矩阵 D 是以 A 的特征值为对角线的元素生成的对角阵，矩阵 A 的第 k 个特征值的特征向量是矩阵 D 的第 k 列列向量，即满足 AV=VD。
- [V，D] = eig(A，'nobalance')　预先不经过平衡处理，得到矩阵 A 的特征值和特征向量。对于一些特定的问题，将矩阵进行平衡处理可以提高所求的特征值和特征向量的精度，然而，如果矩阵有的元素与圆整误差的数量级相接近时，平衡操作将会提高这些小的元素在矩阵中的作用，反而不会得到正确的结果。在这种情况下，可以用'nobalance'来计算。
- d = eig(A，B)　返回矩阵 A 和 B 的广义特征值组成的列向量。对于广义特征值 k 以及相应的特征向量 x，有 Ax=Bxk，x 称为广义右特征向量。A 和 B 必须为方阵。
- [V，D] = eig(A，B)　得到矩阵 A 和 B 的广义特征值矩阵 D 和广义右特征向量组

成的矩阵 V。D 是以广义特征值为对角线元素的对角阵，矩阵 V 的第 k 列是矩阵 D 的第 k 个对角元素的广义右特征向量，并满足 AV=BVD。各个特征向量的范数为 1。

注意：上面的矩阵都应该为方阵，如果在求广义特征值中，矩阵 B 可逆，那么可以转化为求矩阵 inv(B)A 的特征值与特征向量，即 inv(B)Ax=kx。如果矩阵 A 的特征值都不重复，则它是可以对角化的，并且其特征向量组成的矩阵也一定是满秩的，具体内容将在“矩阵对角化”一节中详细介绍。如果函数 eig()的参数为单个的复方阵，则函数相当于 eig(X, eye(X))，函数的求解过程基于 QZ 分解。

例 1

```
A =[  1     4     2
      0    -3     4
      0     4     3]
d=eig(A)
d =[ 1    -5     5]'
[V D]=eig(A)
V =      1.0000    0.4082   -0.6667
              0   -0.8165   -0.3333
              0    0.4082   -0.6667
D =  1     0     0
     0    -5     0
     0     0     5
rank(V)  =   3                                  %矩阵有不同的特征值
[v,d]=eig(A,B)
v =0.7169 - 0.0552i   0.4918 + 0.8427i  -0.5913 - 0.4090i
   0.0436 + 0.3459i   0.0158 + 0.0271i  -0.2120 + 0.2768i
  -0.5477 + 0.2481i   0.1093 + 0.1872i   0.3480 + 0.4903i
d =   0.2494 + 0.5927i          0                  0
        0             0.1680 - 0.0000i        0
        0                   0             0.2494 - 0.5927i
```

5.1.2 特征值求根

利用矩阵的特征值可以方便地求出多项式的根。先用函数 compan(p)求得多项式的友元阵 A，再用函数 eig(A)来求得矩阵 A 的特征值，由线性代数理论可知，矩阵 A 的特征值是其特征多项式的根，而多项式 p 是矩阵 A 的特征多项式。

例 2 求多项式 $x^4-6x^2+3x-8=0$ 的根

```
p =[  1     0    -6     3    -8]
A=compan(p)
A =  0     6    -3     8
     1     0     0     0
     0     1     0     0
     0     0     1     0
v=eig(A)
v = -2.8374
```

```
2.4692
0.1841 + 1.0526i
0.1841 - 1.0526i
```

5.2 矩阵的对角化

矩阵对角化后，矩阵的运算就会变得十分简单，尤其对于维数较多的矩阵，可以大大简化运算，对于手工计算更是如此。在求解常微分方程组时，也要对矩阵进行对角化，矩阵的对角化在实际工程应用中的用处是非常广泛的。几乎所有的对角化都基于特征值与特征向量的求解，特征值与特征向量的求解目的在很大程度上也是为了对角化。

5.2.1 矩阵可对角化的判断

对于 n×n 矩阵 A，由线性代数理论可知，它可以对角化的条件是 A 具有 n 个线性无关的特征向量，也就是对于 A 的每一个特征值，其几何重数要等于它的代数重数，主要基于此来判断矩阵 A 是否可以对角化。

程序代码：

```
function  y=trigle(A)
y=1;c=size(A);
if c(1)~=c(2)                          %判断是否为方阵
   y=0;
   return;
end
e=eig(A);                              %求矩阵的特征值向量
n=length(A);
while 1
   if isempty(e)
      return;
   end
   d=e(1);
   f=sum(abs(e-d)<10*eps);             %找出与 d 相同的特征值的个数
   g=n-rank(A-d*eye(n));               %求 A-d*eye(n)的零空间的秩
   if f~=g                             %如果二者不相等，则矩阵不可对角化
      y=0;return;
   end
   e(find(abs(e-d)<10*eps))=[];  %删除已经判断了的特征值
end
```

对于上述函数，由于计算精度的限制，在进行矩阵秩的计算时有可能出现错误的判断，即出现 rank(A−d×eye(n))增加的现象，在适当情况下可以加大计算的误差。

例 3

```
b=[0 1
    0 0]
```

```
D=[1 2 2
    2 1 2
    2 2 1]
trigle(b) = 0
trigle(D) = 1
```

5.2.2　矩阵的 PAP 对角化

由线性代数理论可知，对于任意可以对角化的矩阵 A，都存在一个可逆矩阵 P，使得 inv(P)AP 为对角阵，对角阵的对角线元素为矩阵 A 的特征值。Matlab 中有可以直接求矩阵 P 的函数 [P，D]=eig(A)，用此函数将矩阵 A 的特征向量矩阵 P 求出，即可以满足上述条件。此矩阵 P 是不唯一的，用上述方法求得的矩阵 P 的列向量长度都为 1，将矩阵 P 的任意列乘以任意非零的实数，所得的矩阵仍然符合条件。可以利用 trigle()函数自己编写函数来求符合条件的可逆矩阵 P，以及对角化后的矩阵。

例 4　[l，k]=eig(D);

```
%由例 3 可知矩阵 D 可以对角化
l(:,1)=4.3*l(:,1);
inv(l)*D*l
ans =
   -1.0000    0.0000    0.0000
    0.0000   -1.0000    0.0000
    0.0000         0    5.0000
```

可知，矩阵 l 可以使矩阵 A 对角化。

复数矩阵的对角化与实数矩阵基本相同，如例 5 所示。

例 5

```
c = [   0.9501 + 0.4447i   0.4860 + 0.9218i   0.4565 + 0.4057i
    0.2311 + 0.6154i   0.8913 + 0.7382i   0.0185 + 0.9355i
        0.6068 + 0.7919i   0.7621 + 0.1763i   0.8214 + 0.9169i]
[v,d]=eig(c);
inv(v)*c*v
ans =
    1.7548 + 2.0680i          0             0.0000 - 0.0000i
     0.0000 - 0.0000i   0.8257 + 0.0511i   0.0000
         0.0000 - 0.0000i   0.0000 + 0.0000i   0.0823 - 0.0193i
```

5.2.3　实对称矩阵的 QRQ 对角化

实对称矩阵 A 都是可以对角化的，并且都存在正交矩阵 Q，使得 inv(Q)AQ 即 Q'AQ 为对角阵，对角阵的对角线元素为矩阵 A 的特征值。对于实对称矩阵，特征值分解函数 eig(A) 返回的特征向量矩阵就是正交矩阵。

例 6

```
F =[  0     1     1    -1
      1     0    -1     1
```

```
    1    -1     0     1
        -1     1     1     0]
[fd,fv]=eig(F);
fd'*fd
ans = 1.0000    0.0000    0.0000    0.0000
      0.0000    1.0000    0.0000    0.0000
      0.0000    0.0000    1.0000    0.0000
      0.0000    0.0000    0.0000    1.0000
fd'*F*fd
ans =1.0000   0.0000   0.0000   0.0000
     0.0000   1.0000   0.0000   0.0000
     0        0        -3.0000  0.0000
     0.0000   0.0000   0.0000   1.0000
```

5.3 Jordan 标准形与矩阵相似

如果两个矩阵 A 和 B 相似，那么存在可逆矩阵 P，使得 A＝inv(P)BP。在 MatLab 中，判断两个矩阵相似比较困难，没有现成的函数来求两个矩阵 A 和 B 的过渡矩阵 P，使得 A＝inv(P)BP，也没有判断矩阵相似的函数。由线性代数理论可知，n 阶方阵都与一个 Jordan 标准形相似，并且这个 Jordan 标准形是唯一的，相似矩阵都与同一个 Jordan 标准形相似。那么判断矩阵相似最有效的方法就是，第 1 步，找出与矩阵相似的 Jordan 标准形。在实际应用过程中，有一套求矩阵的 Jordan 标准形和其过渡矩阵的方法，并以函数形式给出，附有必要的说明；第 2 步，利用矩阵的 Jordan 标准形，给出判断矩阵相似的函数。

5.3.1 Jordan 标准形的计算

虽然 MatLab 中还没有直接求出矩阵的 Jordan 标准形的函数，但可以利用 MatLab 强大的矩阵处理功能，直接计算矩阵的 Jordan 标准形。为了便于求解 Jordan 标准形，更好地理解用 MatLab 求解 Jordan 标准形的过程，给出了一个求矩阵的 Jordan 标准形的函数，jordan()。此过程共包含 isjordan(A)，diaglink(A，B)，lk(k，n)，[k ，kn，kr]=cc(A)，[p，y]=jordan(A)5 个函数，函数的源代码、参数说明及功能如下。

1. besquare ()函数

此函数用来判断矩阵 A 是否可以构成 Jordan 块，即是否为方阵。此函数并没有判断除 0－对角线及 1－对角线元素的其他元素是否为非零值，因为这些元素在以后的连接过程中将被忽略，因而意义不大。

```
function y=besquare (A)
c=size(A);
nx=c(1);ny=c(2);
y=1;
if nx~=ny                              %判断是否为方阵
   y=0;
```

```
    return;
  end
```

2. diaglink()函数

此函数用以连接两个Jordan块A和B，函数返回连接成功后的矩阵。函数只抽取矩阵A和B的0—对角线元素和1—对角线元素并分别加以连接，以构成返回矩阵的0—对角线和1—对角线，完成连接。不在这两条对角线上的其余元素将被忽略。

```
function y=diaglink(A,B)
if ~besquare(A)|~ besquare(B)                    %判断是否可以构成jordan阵
   error('The matrix you input is not a Jordan one');
   return;
end
if isempty(A)
  if length(B)>1                                 %删除B中不符条件的元素
     y=diag(diag(B))+diag(diag(B,1),1);
  else
     y=B;
  end
elseif isempty(B)
  if length(A)>1                                 %删除A中不符条件的元素
     y=diag(diag(A))+diag(diag(A,1),1);
  else
     y=A;
  end
else
     da=diag(A)';                                %提取A的0—对角线元素
     db=diag(B)';                                %提取B的0—对角线元素
     da1=diag(A,1)';                             %提取A的1—对角线元素
     db1=diag(B,1)';                             %提取B的1—对角线元素
     if length(A)==1
        da1=[];
     end
     if length(B)==1
        db1=[];
     end
     d=[da db];                                  %连接0—对角线元素
     d1=[da1 0 db1];                             %连接1—对角线元素
     c1=diag(d);                                 %生成0—对角线矩阵
     c2=diag(d1,1);                              %生成1—对角线矩阵
     y=c1+c2;                                    %合成结果矩阵
 end
```

例 7

```
1 =[ 1    1    1
     0    2    1
```

```
       1     3     1]
diaglink(1,1)
ans =
         1     1     0     0
         0     2     1     0
         0     0     1     0
         0     0     0     1
diaglink(1,1)
ans =    1     0
               0     1
```

3. lk()函数

此函数用来生成特征值 k 的 n 阶 Jordan 块，n 应该为整数，且大于 0。虽然函数增加了将 n 自动转化为整数的功能，但在实际计算过程中 n 不应该出现小数，否则有可能使程序计算出错。

```
function y=lk(k,n)
n=fix(n);                                  %圆整
if n<1
   error('the argument n should be great than 0');
   return;
end
if n==1
   y=k;
elseif n>1
      c=diag(ones(1,(n-1)),1);%生成 1一对角矩阵
      d=k*eye(n);                          %生成 0一对角矩阵
      y=c+d;                               %合成结果矩阵
   end
```

例 8

```
lk(4,3)
ans =
        4     1     0
        0     4     1
        0     0     4
```

4. lkk()函数

[cc，y]=lkk(A，k，v)　此函数生成矩阵 A 关于特征值 k 的所有 Jordan 块并加以连接，同时返回特征值 k 的特征向量和广义特征向量构成的矩阵 cc，此矩阵用以构成最终的可逆矩阵 P，以使矩阵 A 化成 Jordan 标准形。Y 是 k 的 Jordan 块连接而成的矩阵。V 是矩阵 A 的特征值 k 的 Jordan 块的说明向量，如 v(i)，下标 i 表示 Jordan 块的重数，而 v(i)的值为整数，表示当前重数的 Jordan 块的个数，如 v(2)=2，表示二重 Jordan 块的个数为 2。在函数执行中，先求矩阵 A 关于特征值 k 的基特征向量矩阵 D，然后利用循环来求各阶 Jordan 块的广义特征

向量，并将基特征向量和广义特征向量按顺序连接成特征向量矩阵，便于生成可逆矩阵 P。

```
function [cc,y]=lkk(A,k,v)
n=length(v);
AA=A-k*eye(length(A));                %得到 k 的特征矩阵
D=null(AA);                           %得到特征矩阵的零空间
kk=1;
cc=[];
y=[];
for i=1:n
   for j=1:v(i)
      cc=[cc D(:,kk)];                %提取基特征向量
      b=D(:,kk);
      for l=2:i
         x=pinv(AA)*b;                %求广义特征向量
         cc=[cc x];                   %生成广义特征向量矩阵
         b=x;
      end
      kk=kk+1;
      c=lk(k,i);                      %生成 Jordan 块
      y=diaglink(y,c);                %连接 Jordan 块
   end
end
```

例 9

```
A =
     1     2     2     0     0
     2     1     2     0     0
     2     2     1     0     0
     0     0     0     5     0
     0     0     0     2     5
[cc y]=lkk(A,5,[1 1])
cc =
    0.5774         0         0
    0.5774         0         0
    0.5774         0         0
         0         0    0.5000
         0    1.0000         0
y = 5     0     0
          0     5     1
          0     0     5
```

可见，例 9 中生成了一个 1 重 Jordan 和一个二重矩阵。

5. cc()函数

[ks，kn，kr]=cc(A)　此函数用以生成矩阵 A 的特征值向量 ks、特征值的代数重数组成的向量 kn，即特征值 ks(i)的代数重数 kn(i)和几何重数 kr(i)。在 MatLab 中，由于计算误差

的存在，本来特征值应该为实数，但计算所得的特征值含有虚部，通常虚部都很小，可以用 real(e)来将虚部删掉。在求解过程中，先将所有的特征值排序，这是为了以后相似矩阵的判断，这样，生成的 Jordan 标准形的特征值都是从大到小排列或从小到大排列，从而保证了在相似矩阵的 Jordan 标准形中，各 Jordan 块的排列顺序是完全一样的，便于判断。然后进行特征值分解，将特征值的虚部删掉。从特征值向量的第 1 个特征值进行分析，变量 f 保存原有的特征值向量，便于提取相对应的特征向量。先用 sum()函数统计特征值向量中，所有与第 1 个特征值相同的特征值的个数，组成此特征值的代数重数，并由此特征值的零空间的秩来求得其几何重数，然后将这些特征值从特征值向量中删除。不断重复这一过程，直到特征值向量为空，即所有的特征值都已经进行了判断。

```
function [ks,kn,kr]=cc(A)
[v e]=eig(A);                                %特征值分解
e=diag(e);                                   %求特征值向量
ne=length(e);
for i=1:ne-1                                 %将特征值排序
   for j=i+1:ne
      if e(i)>e(j)
         temp=e(i);e(i)=e(j);e(j)=temp;
      end
   end
end
e=real(e)                                    %删除由于计算误差引起的虚部
f=e;
n=length(A);
ks=[];kn=[];kr=[];
while ~isempty(e)
   c=e(1)
   ks=[ks c];                                %连接特征值向量
   s=sum(abs(e-c)<0.0001);                   %求特征值的代数重数
   F=v(:,find(abs(f-c)<0.0001))
   r=rank(F);                                %求矩阵的几何重数
   kn=[kn s];                                %连接代数重数向量
   kr=[kr r];                                %连接几何重数向量
   e(find(abs(e-c)<0.0001))=[];              %删除处理后的矩阵
end
```

例 10

```
[ks kn kr]=cc(A)
ks =  -1.0000    5.0000
kn = 2     3
kr = 2     2
```

6. jordan()函数

此函数是生成 jordan 矩阵的主要函数，本函数的核心部分是生成特征值的 jordan 块的说明向量 v。返回参数 P 是将矩阵 A 转化为 jordan 标准形的可逆过渡矩阵，而 y 则是 A 的

Jordan 标准形。在求特征值说明向量 v，即求特征值的各重 jordan 块的数目，向量中各元素的值的和应该为相应特征值的几何重数，而 v 中各元素与其下标的乘积的和则为此特征值的代数重数。求各重 jordan 块的数目时，先从二重矩阵开始，利用公式 v(j)=rank(RA^(j-1))+rank(RA^(j+1))-2rank(RA^j)来求解，并进行循环，如果剩余的特征值的几何重数等于其剩余的代数重数，则说明剩余的所有的 jordan 块都为一重 jordan 块，其数目也即等于剩余的几何重数或代数重数。

```
function [p,y]=jordan(A)
if ~issquare(A)                              %判断是否为方阵
   error('The Matrix must be square');
   return;
end
y=[];n=length(A);
p=[];
if n==0                                      %判断是否为空阵
   error('The matrix should not be empty');
   return;
end
if n==1                                      %判断矩阵 A 是否为常数值
   p=1;
   y=A;
   return;
end
y=[];n=length(A);
p=[];
[ks kn kr]=cc(A);                            %求 A 的特征值向量、几何和代数重数向量
for i=1:length(ks)                           %对特征值进行循环
   v=[];
   if kn(i)==1                               %如果代数重数为 1 则为一个一阶 jordan
      v(1)=1;                                %块
   elseif kr(i)==1                           %如果几何重数为 1，则为一个 kn(i)阶
      v(kn(i))=1;                            %jordan 块
   elseif kn(i)==kr(i)                       %代数重数等于代数重数，则为 kn(i)
      v(1)=kn(i);                            %个一阶 jordan 块
   else
      v(1)=0;
      RA=A-ks(i)*eye(n);                     %求特征矩阵
      j=2;
      while 1
         v(j)=rank(RA^(j-1))+rank(RA^(j+1))-2*rank(RA^j);
          %求各阶 Jordan 的块数
         ss=0;rr=0;
         for ii=2:j
            ss=ss+ii*v(ii);                  %现有的代数重数的和
            rr=rr+v(ii);                     %现有的几何重数的和
         end
```

```
            aa=kn(i)-ss;                              %剩余的代数重数
            if aa==(kr(i)-rr)                         %剩余的代数重数等于剩余的几何重数
               v(1)=aa;
               break;
            end
            j=j+1;
         end
      end
      [cc AA]=lkk(A,ks(i),v);                         %求 jordan 块及特征向量矩阵
      p=[p cc];                                       %连接特征向量矩阵
      y=diaglink(y,AA);                               %连接 jordan 块
   end
```

例 11 [yy c]=jordan(A)

```
yy =
   -0.8165         0    0.5774         0         0
    0.4082   -0.7071    0.5774         0         0
    0.4082    0.7071    0.5774         0         0
         0         0         0         0    0.5000
         0         0         0    1.0000         0
c =    -1.0000         0         0         0         0
             0   -1.0000         0         0         0
             0         0    5.0000         0         0
             0         0         0    5.0000    1.0000
             0         0         0         0    5.0000
inv(yy)*A*yy
ans =
    -1.0000         0    0.0000         0         0
     0.0000   -1.0000    0.0000         0         0
          0    0.0000    5.0000         0         0
          0         0         0    5.0000    1.0000
          0         0         0         0    5.0000
H =
     3     0     2    -1
     0     3    -2     2
     0     0     3     0
     0     0     0     3
```

说明：特征值是为实数的实矩阵设计的，而对于特征值为复数的实矩阵和复数对角阵，只要对函数的某些部分作适当的修改即可以满足要求。实践证明，对于元素为整数的矩阵，和某些计算精度较好的小数矩阵，用函数完全可以得到正确的结果。但是对于另一些矩阵，一个主要因素是由于在函数的嵌套调用过程中某些数值的传递精度会降低，在特征值分解的过程中本应该得到实特征值和特征向量，却得到了复特征值和特征向量，而本程序是按实特征值和特征向量设计的；另一个主要因素，在函数内部许多地方用到矩阵求秩的函数，在计算过程中产生的误差超出了计算机求秩的容许误差，从而造成求秩的错误，一般是所求得矩阵的秩高于其实际的秩。一条解决的途径是加大求秩函数的计算误差，而最根本的

途径就是寻求得到一种能准确求解特征矩阵的秩的有效的方法，代替上面的求秩函数即可。另外，在 MatLab 中的工作空间中对特征矩阵直接求秩也可以得到正确的结果。

5.3.2　相似矩阵的判断

如果两个矩阵 A 和 B 相似，则存在可逆矩阵 P，使得 A＝inv(P)BP，相似矩阵有相同的特征值，各个特征值的重数也相同，这是矩阵相似的必要条件；矩阵相似的重要条件是矩阵 A 和 B 具有相同的 jordan 标准形，标准形中各 jordan 块的顺序则无关。

下面利用程序 jordan()来编写一个判断两个矩阵是否相似的函数，如果两个矩阵相似，则得到两个矩阵的过渡矩阵 P，使得 A=inv(P)BP。函数先判断两个矩阵的所有特征值是否完全相同，如果相同，再利用 jordan()函数求两个矩阵的 jordan 标准形及相应的过渡矩阵，然后利用两个过渡矩阵求 A 到 B 的过渡矩阵。

```
function [y,p]=alikeb(A,B)
y=1;p=[];                                    %预先赋值
if ~besquare(A)|~besquare(B)                 %判断矩阵是否为方阵
   y=0;
   error('The Matrixes should be square');
   return;
end
na=length(A);
nb=length(B);
if na~=nb                                    %判断两个矩阵大小是否相同
   y=0;
   return;
end
n=na;
ea=eig(A);                                   %求 A 的特征值向量
ea=real(ea);                                 %删去虚部
eb=eig(B);                                   %求 B 的特征值向量
eb=real(eb);                                 %删去虚部
for i=1:n-1                                  %对两个矩阵的特征值排序
   for j=i+1:n
      if ea(i)>ea(j)
         temp=ea(i);ea(i)=ea(j);ea(j)=temp;
      end
      if eb(i)>eb(j)
         temp=eb(i);eb(i)=eb(j);eb(j)=temp;
      end
   end
end
if ~all(abs(ea-eb)<10*eps)              %判断两个特征值向量是否相同
   y=0;
   return;
end
[ya ja]=jordan(A);                      %求 A 矩阵的 jordan 标准形和过渡矩阵
```

```
[yb jb]=jordan(B);                      %求A矩阵的 jordan 标准形和过渡矩阵
if ~all(abs(ja-jb)<10*eps)              %判断两个 jordan 标准形是否相同
   y=0;
   return;
end
p=yb*inv(ya);                           %求A到B的过渡矩阵 P
```

例 12

```
h =
     3     0     4     1
     0     3    -2     0
     0     0     3     0
     0     0     0     3
[y p]=alikeb(H,h)
    y =  1
p =
       1.0000         0         0         0
            0   -1.0000    0.0000    0.0000
            0    0.0000   -1.0000    1.0000
            0    0.0000    6.0000   -5.0000
inv(p)*h*p
ans =
    3.0000    0.0000    2.0000   -1.0000
         0    3.0000   -2.0000    2.0000
         0    0.0000    3.0000    0.0000
         0    0.0000    0.0000    3.0000
```

说明：此函数也应该满足 jordan()函数所需要的条件，即要求函数中的求秩运算和求特征向量的运算必须正确，且为实矩阵，否则会出错。如果在程序运行过程中出现"矩阵下标超出矩阵的维数"的错误，则一定是在函数中的求秩的运算出现了错误，致使程序中特征值的代数重数、几何重数以及 jordan 块的数目等之间的数量关系发生了混乱。如果上面的错误没有发生，而所求得的过渡矩阵 P 使得 inv(P)BP 与矩阵 A 有很大的偏差，则主要是因为矩阵 A 和 B 中求得的特征向量矩阵是接近奇异的，或是矩阵进行特征值分解时由于计算误差出现了复数特征向量和复数特征值，而又将特征值的虚部删去的原因。

5.4 一元多项式的运算

一元多项式在代数中占有非常重要的地位。在实际应用中，如对实验数据的插值、微商和曲线拟合等，都要大量用到多项式；在矩阵分析时，也要用到一元多项式的概念。多项式函数是形式最简单的函数，也是最容易计算的函数，从理论上讲，它可以表示绝大多数复杂函数。在许多计算机的计算和编程中，很多函数值如 sin(x)，cos(x)等的计算都是先将函数进行 Tailor 展开为多项式进行逼近计算的，并且都能达到很高的精度。下面介绍 MatLab 中的多项式的表示和各种运算。

5.4.1 多项式的表示和创建

任意多项式都可以用一个行向量来表示，即 n+1 维的向量 a 表示多项式 y(x)=a(1)*x^n+a(2)*x^(n-1)+…+a(n)*x+a(n+1)，可以看出 n+1 维的向量表示阶数为 n 的多项式。且任意一个向量就可以作为多项式。

例 13

```
p=[1    -6    11    -6]
poly2sym(p,'x')
    ans =x^3-6*x^2+11*x-6
```

1. poly()函数

p=poly(A)　A 是一个 n×n 的矩阵，此函数返回矩阵 A 的特征多项式 p，p 是 n+1 维向量，特征多项式的根就是矩阵 A 的特征值。

例 14

```
A =
     6   -11     6
     1     0     0
     0     1     0
p =    1.0000   -6.0000   11.0000   -6.0000
roots(p)                            %求多项式的根
ans =    3.0000    2.0000    1.0000
eig(A)                              %求 A 的特征值
ans =    3.0000    2.0000    1.0000
polyvalm(p,A)                       %求 A 关于多项式 p 的值
ans =
  1.0e-013 *                        %结果应该为 0，存在计算误差
   -0.1066    0.2132   -0.1421
   -0.0178         0         0
         0   -0.0178         0
```

可以利用例 14 来求矩阵的特征值。

事实上，由代数理论可以知道，矩阵关于其特征多项式的值为 0。

p=poly(r)　r 为向量，此函数返回以向量中的元素为根的多项式。

例 15

```
v=[1 2 3]
p=poly(v)
p =   1    -6    11    -6
```

2. 多项式的符号表示

poly2sym(p)　函数将向量表示的多项式表示为符号表示的多项式。

poly2sym(p)　用字符'x'来表示多项式的变量。而 poly2sym(p，'v')和 poly2sym(p，sym(v))

则用字符'v'来表示多项式 p 中的变量。

例 16

```
p=[2 4 -1 3];
poly2sym(p)
    ans =2*x^3+4*x^2-x+3
poly2sym(p,'s')
ans =2*s^3+4*s^2-s+3
```

5.4.2 多项式的基本运算

1. 多项式的加减运算

MatLab 内没有给出多项式的加减运算，而多项式的加减运算是统一的，编程很容易实现，下面给出两个多项式加法运算的函数。

```
function yy=pplus(x,y)
nx=length(x);
x=reshape(x,1,nx);
ny=length(y);
y=reshape(y,1,ny);
n=max(nx,ny);
cc=zeros(1,n);
if nx>ny
   cc(1,(nx-ny+1):nx)=y;
   yy=x+cc;
elseif nx<ny
   cc(1,(ny-nx+1):ny)=x;
   yy=y+cc;
else
   yy=x+y;
end
```

例 17 已知：p1=[3 4 6]，p2=[5 2 -4 7]，求两个多项式的和与差

```
p3=pplus(p2,p1)
p3 =
5     5     0    13
p3=pplus(p2,-p1)
p4 =
 5    -1    -8     1
```

2. 多项式的乘法运算

w=conv(u，v) 此函数求多项式 u 和 v 的乘积，即求向量 u 和 v 的卷积。如果 m=length(u)，n=length(v)，则 w 的长度为 m+n-1，并且满足 w(k)=a(1)b(k)+a(2)b(k+1)+…+a(k)b(2k-1)。

例 18　求 p1 和 p2 的乘积

```
p=conv(p1,p2)
p =
 15    26    26    17     4    42
poly2sym(p)
ans =
15*x^5+26*x^4+26*x^3+17*x^2+4*x+42
```

3. 多项式的除法运算

[q，r] = deconv(v，u)　此函数表示多项式 v 除以多项式 u 得到商多项式 q 和余数多项式 r，如果 r 的元素全部为零，则表示多项式 v 可以整除多项式 u。多项式运算实际上是进行解卷积的运算。

例 19　求多项式 p 除以多项式 p1

```
 [py,r]=deconv(p,p1)
py =
    5.0000    2.0000   -4.0000    7.0000
r =
  1.0e-014 *                    %余数多项式接近零
       0         0         0    0.3553    0.3553    0.7105
```

4. 求多项式的根

roots(c)　此函数返回多项式的根组成的向量，同时也是多项式的友元阵的特征值向量。对于系数为实数的多项式的根，则如果其根出现复数，则复数必是成对出现的。

例 20

```
roots(p)
ans =
  -1.5257
  -0.6667 + 1.2472i
  -0.6667 - 1.2472i
   0.5629 + 0.7751i
   0.5629 - 0.7751i
B=compan(p)
B =
   -1.7333   -1.7333   -1.1333   -0.2667   -2.8000
    1.0000         0         0         0         0
         0    1.0000         0         0         0
         0         0    1.0000         0         0
         0         0         0    1.0000         0
eig(B)
ans =
  -1.5257
  -0.6667 + 1.2472i
  -0.6667 - 1.2472i
```

```
0.5629 + 0.7751i
0.5629 - 0.7751i
```

5. 多项式的数组运算

y = polyval(p，x) 计算多项式在 x 处的值，x 可以是矩阵或向量，此时函数计算多项式在 x 的每个元素处的值。

例 21

```
C =
    0.9501    0.4860    0.4565
    0.2311    0.8913    0.0185
    0.6068    0.7621    0.8214
polyval(p,C)
ans =
  116.2514   52.8001   51.2670
   44.2379  102.3256   42.0800
   61.2585   79.0563   88.6102
```

6. 多项式的矩阵运算

Y = polyvalm(p，X) p 是多项式的系数向量，相当于用矩阵 X 代替多项式的变量来对矩阵而不是对数组进行计算，X 必须为方阵。

例 22

```
polyvalm(p,C)
ans =
  282.6905  290.8218  149.6809
   83.8361  169.5294   42.8069
  256.9352  332.8976  206.7038
```

7. 部分分式展开(留数计算)

函数 residue()在部分分式展开和多项式系数之间进行转换。

[r，p，k] = residue(b，a) 找出两个多项式 b(x)和 a(x)之比 b(x)/a(x)的留数、极点及直项向量，分别用 r，p，k 来表示。

[b，a]=residue(r，p，k) 将部分分式展开的形式还原为两个多项式 b(x)和 a(x)相除的形式。

如果多项式 a(x)不含重根，则两个多项式之比可以写成 b(x)/a(x)=r1/(x-p1)+r2/(x-p2)+…+rn/(x-pn)+k(x)，其中，p1，p2，p3，…，pn 称为极点，r1，r2，r3，…，rn 称为留数，k(x)称为直项。留数和极点的个数 n 满足 n=length(a)-1=length(r)=length(p)。如果 b 的次数小于 a 的次数，则直项的系数向量的系数为空；否则，它们之间满足 length(k)=length(b)-length(a)+1。

如果 a(x)含有 m 重根 p(j)= …=p(j+m-1)，则展开的这 m 项应该写成 r(j)/(x-pj)+r(j+1)/$(x-pj)^2$+ …+r(j+m-1)/$(s-pj)^m$。

说明：本函数是一个 M 文件，求解的过程大体是这样的：先用多项式求根公式 roots()求出各个极点，再用多项式的长除法 deconv()求出直项 k，最后计算留数。对于重根的情况，用 resi2.m 来计算重根处的留数。注意：如果分母多项式 a(x)是接近具有重根的多项式，则多项式系数即使在圆整误差范围内的小的变化也会对极点及留数的计算结果造成很大的影响。

例 23

```
a=[2 4 6 8 1]
b=[1 3 9 2]
[r,p,k]=residue(b,a)                 %将多项式除部分分式展开
r =
   0.6935
  -0.1587 - 0.5193i
  -0.1587 + 0.5193i
   0.1239
p =
  -1.5780
  -0.1420 + 1.5082i
  -0.1420 - 1.5082i
  -0.1381
k = []
[b1,a1]=residue(r,p,k)               %将部分分式展开还原
b1 =
    0.5000    1.5000    4.5000    1.0000
a1 =
    1.0000    2.0000    3.0000    4.0000    0.5000
```

8. 微分多项式

函数 polyder()可以用来计算多项式、多项式相乘及多项式相除的微分。

k = polyder(p)　得到多项式 p 的微分多项式 k。

k=polyder(a，b)　得到两个多项式 a 和 b 乘积的微分多项式 k，即相当于 k=polyder(conv(a，b))。

[q，d] = polyder(b，a)　得到两个多项式的商 b/a 的微分，其中 q 是结果的分子多项式，d 是结果的分母多项式。

例 24

```
polyder(a)
ans =
     8    12    12     8
polyder(a,b)
ans =
    14    60   180   264   261   174    25
     [q,d]=polyder(b,a)
q =
```

```
      -2   -12   -60   -72   -51   -18    -7
d =
       4    16    40    80   104   104    76    16     1
```

5.4.3　多项式的因式分解

对于 n 阶式系数或复系数多项式，都有 n 个根 r_1，r_2，r_3，…，r_n，则对多项式在复数范围内进行因式分解就非常简单，分解后的各个因式就是 $x-r_1$，$x-r_2$，…，$x-r_n$。对于实系数多项式在实数范围内进行因式分解，如果对多项式求根，则有可能产生复根，要将复因式合并，就会比较复杂。由线性代数的理论可知，实系数多项式的根是成对出现的，且其中两对根之间互为共轭，在实数范围内因式分解后的因式的最高次数不会超过 2。下面给出对实系数多项式在实数范围内进行因式分解的函数，返回一个 n×3 的矩阵，n 为分解后的因子个数，矩阵的每行就代表一个因子。

```
function y=expoly(p)
s=p(1);
r=roots(p);                                  %求多项式的根
y=[];
while ~isempty(r)
   c=r(1);
   r(1)=[];                                  %删除此根
   if imag(c)>eps                            %此根为复数
      cc=conv([1 -c],[1 -conj(c)]);%合并复数因子
      y=[y                                   %连接结果矩阵
         cc];
      r(find(abs(r-conj(c))<eps))=[];        %删除与其共轭的根
   else                                      %根为接近实数
      c=real(c);                             %取其实部
      cc=[0 1 -c];                           %生成因式
      y=[y                                   %连接
         cc];
   end
end
if abs(s-1)>eps                              %多项式最高项系数不为 1
   cc=[0 0 s]                                %生成常数因子
   y=[y
      cc];
end
```

例 25

```
p =
     1     2     3     4     5
c=expoly(p)
c =
```

```
    1.0000   -0.5756    2.0882
    1.0000    2.5756    2.3945
conv(c(1,:),c(2,:))
ans =
1.0000    2.0000    3.0000    4.0000    5.0000
```

5.4.4 最大公因式和最小公倍式

最大公因式和最小公倍式是整系数多项式的两个比较重要的概念，MatLab本身并没有提供用来求多项式的最大公因式和最小公倍式的函数，为了方便对多项式的操作，特意编写了两个函数用来完成这两个操作，仅供参考。

1. 最大公因式

对于2个多项式的最大公因式，利用循环相除就可以很容易求得，利用多项式求根进行判断也是可以的。下面给出循环相除法的代码：

```
function y=gcd(f,g)
  while ~isempty(f)                         %删除 f 的前导零
     if abs(f(1))<2*eps
        f(1)=[];
     else
        break;
     end
  end
  while ~isempty(g)                         %删除 g 的前导零
     if abs(g(1))<2*eps
        g(1)=[];
     else
        break;
     end
  end
  if isempty(f)|isempty(g)                  %f 和 g 不应该为空
     error('Arguments should not be null');
     return;
  end
  f=f/f(1);g=g/g(1);                        %将最高项系数化为 1
while 1
  [q,r]=deconv(f,g);                        %多项式长除
  while ~isempty(r)                         %删除 r 的前导零
     if abs(r(1))<2*eps
        r(1)=[];
     else
        break;
     end
  end
  if isempty(r)                                  %r 为空则表示已经整除
```

```
      y=g/g(1);                                    %将 y 的最高项系数化为 1
      return;
   else                                            %为下一步循环作准备
      f=g;
      g=r;
   end
   if length(r)==1                                 %r 如为常数，表示公因子为 1
      y=1;
      return;
   end
end
```

例 26

```
gcd([1 1 -2],[1 2 -3])
ans =    1    -1
p1 =    1    10    31    30
p2 =    1    12    41    42
gcd(p1,p2)
ans =    1     5     6
```

2. 最小公倍式

在求得最大公因式后，将两个多项式的乘积除以二者的最大公因式即可以得到最小公倍式。

```
function y=lcd(f,g)
r=gcd(f,g);
y=deconv(conv(f,g),r);
```

例 27 rp=lcd(p1，p2)

```
rp =
 1    17   101   247   210
```

5.5 矩阵的分解

矩阵分解是一个非常重要的概念，如求矩阵的特征值和特征向量、矩阵的秩等重要参数时都要用到矩阵的分解。在实际工程中，在特定的场合要对矩阵进行特定形式的分解。在 MatLab 中，有许多现成的公式可以利用，因而比较容易进行矩阵分解。本节主要介绍以下几种矩阵分解形式。

5.5.1 实对称正定矩阵的cholesk 分解

R=chol(X) 此函数只使用矩阵 X 的对角和上三角元素。而下三角元素被认为是上三角元素的转置，产生一个上三角矩阵 R 使得 R'R=X，X 必须为对称且是正定的，如果矩阵非正定，则会打印出错误信息。

使用[R，p]=chol(X)不会打印出任何错误信息，如果X是正定的，则p=0R与chol(X)的结果相同；如果X不是正定的，则p是一个正整数，R是一个q=p-1阶的上三角阵，使得R'R=X(1:q，1:q)。

例 28

```
X=pascal(4)
X =
  1     1     1     1
     1     2     3     4
     1     3     6    10
     1     4    10    20
R=chol(X)
R =
 1     1     1     1
     0     1     2     3
     0     0     1     3
     0     0     0     1
R'*R
ans =
     1     1     1     1
     1     2     3     4
     1     3     6    10
     1     4    10    20
```

对于此分解，可以自编函数加以实现，下面给出另一个函数来实现此操作。计算中采用的公式为：L(k，k)=sqrt[A(k，k)-(L(k，1)^2+L(k，2)^2+…+L(k，k-1)^2)]和L(j，k)=[A(j，k)-(L(j，1)L(k，1)+L(j，2)L(k，2)+ …+L(j，k-1)L(k，k-1))]/L(k，k)。下面给出程序的代码。

```
function y=llt(A)
if ~isequal(A,A')                                    %判断矩阵是否为对称
   error('A must be symtrix');
   return;
end
n=length(A);
e=eig(A);
if ~all(e>0)                                         %判断矩阵是否正定
   error('A must be positive');
   return;
end
l(1,1)=sqrt(A(1,1));                                 %求第1列数据
l(2:n,1)=A(2:n,1)/l(1,1);
for i=2:n
   sum=0;
   for j=1:(i-1)
      sum=sum+l(i,j)^2;
   end
```

```
    l(i,i)=sqrt(A(i,i)-sum);          %求对角线元素的值
    for j=(i+1):n
       sum=0;
       for k=1:(i-1)
          sum=sum+l(j,k)*l(i,k);
       end
       l(j,i)=(A(j,i)-sum)/l(i,i);    %求同列其余元素的值
    end
 end
 y=l';                                %返回 l 的转置
```

例 29

```
b=llt(X)
b =
     1     1     1     1
     0     1     2     3
     0     0     1     3
     0     0     0     1
```

5.5.2 实对称正定矩阵的 ldl 分解

对于实对称正定矩阵 A，都存在下三角矩阵 L 和对角阵 D，使 A=LDL'，且 L 和 D 都唯一。程序代码如下：

```
function [l,d]=ldl(A)
%参数 A 为对称正定矩阵，否则会出错，返回参数 l 为得到的下三角矩阵
%d 为对角阵。
if ~isequal(A,A')
   error('Matrix must be symetric!');
   return;
end
n=length(A);
v=eig(A);
if ~all(v>0)
   error('Matrix must be positive!');
   return;
end
w(1)=A(1,1);
for i=1:n
   l(i,i)=1;
end
l(2:n,1)=A(2:n,1)/w(1);
for i=2:n
   s=0;
   for kk=1:(i-1)
      s=s+l(i,kk)^2*w(kk);
   end
   w(i)=A(i,i)-s;
```

```
      for j=(i+1):n
         s=0;
         for kk=1:(i-1)
            s=s+l(j,kk)*l(i,kk)*w(kk);
         end
         l(j,i)=(A(j,i)-s)/w(i);
      end
   end
      d=diag(w);
```

例 30

```
E = 5       -4       1
       -4       6       -4
       1       -4       6
[l,d]=ldl(E)
l =
    1.0000          0              0
    -0.8000         1.0000         0
     0.2000         -1.1429        1.0000
d =  5.0000         0              0
    0               2.8000         0
    0               0               2.1429
l*d*l'
ans =     5    -4     1
         -4     6    -4
          1    -4     6
```

5.5.3 矩阵的 lu 分解

矩阵的 lu 分解很重要，许多运算都是以 lu 分解为基础的，如方阵的求逆操作 inv()、行列式操作 det()，它也是求解线性方程组即除法运算的基础。lu 分解采用高斯消去法将矩阵分解为两个三角矩阵的乘积。

[L，U] = lu(X) 此函数得到一个上三角矩阵 U 和一个准下三角矩阵(即下三角矩阵的转置矩阵)，使得 X=LU。

[LL，U，P] = lu(X) 此函数得到一个上三角矩阵 U、一个下三角矩阵 LL 和一个置换矩阵 P，使得 LLU=PX。可知，L=inv(P)LL。

例 31

```
s =
    1   1   1   1
    2   2   2   2
    3   2   4   5
    6   7   9   0
[L,U]=lu(s)
L = 0.1667  0.1111  0.5000  1.0000
    0.3333  0.2222  1.0000  0
```

```
     0.5000  1.0000  0       0
     1.0000  0       0       0
U =  6.0000  7.0000  9.0000  0
     0       -1.5000 -0.5000 5.0000
     0       0       -0.8889 0.8889
   . 0       0       0       0
[ll,u,p]=lu(s)
ll = 1.0000 0       0       0
     0.5000 1.0000  0       0
     0.3333 0.2222  1.0000  0
     0.1667 0.1111  0.5000  1.0000
u =  6.0000 7.0000  9.0000  0
     0      -1.5000 -0.5000 5.0000
     0      0       -0.8889 0.8889
     0      0       0       0
p = 0       0       0       1
    0       0       1       0
    0       1       0       0
    1       0       0       0
inv(p)*ll
ans = 0.1667    0.1111  0.5000  1.0000
      0.3333    0.2222  1.0000  0
      0.5000    1.0000  0       0
      1.0000    0       0       0
```

说明：在求矩阵 A 的行列式的值时，先对矩阵 A 进行 lu 分解，生成矩阵 L 和 U，然后用 L 和 U 的对角线元素的乘积来计算矩阵 A 的行列式的值。在求解除法运算 x=A\b 时，也是先对矩阵 A 进行 lu 分解，然后分别求解 y=L\b 和 x=U\y，从而得到 x。

5.5.4 矩阵的 qr 分解

qr 分解即矩阵的正交三角分解，此分解适用于任意的矩阵，是非常重要的分解形式。它将矩阵分解为一个正交矩阵和一个上三角矩阵的乘积。

[Q，R] = qr(X)　此函数得到一个与 X 同阶的上三角阵 R 和一个酉矩阵 Q 使得 X=QR。

[Q，R，E] = qr(X)　此函数得到一个置换矩阵 E、一个对角线元素递减的上三角阵 R 和一个酉矩阵 Q，使得 XE=QR。选择置换矩阵 E 使得 abs(diag(R))递减。

A = qr(X)　此函数得到一个 LINPACK 的 ZQRDC 程序的输出，TRIU(qr(X))的结果是 R。

例 32　[q，r，e]=qr(s)

```
q = -0.0990 -0.1545 0.4078  -0.8944
    -0.1980  -0.3090 0.8157  0.4472
    -0.3961  -0.8227     -0.4078 0.0000
    -0.8911  0.4515      -0.0453 0.0000
r = -10.0995     -2.4754     -7.5251 -7.0300
     0           -4.8860 0.7424  -0.5317
```

```
      0          0          0.9063  0.5438
      0          0          0       0.0000
e =   0     0    0    1
      0     0    1    0
      1     0    0    0
      0     1    0    0
```

可以验证例32中的q为一个酉矩阵。

qr分解用来解方程个数多于未知数个数的线性方程组，如例33所示：

例 33

```
A =[  1     1     1
      2     2     2
      3     3     3
          3     7     9]
b =[   1     2     3     4]'
```

求解 x=A\b 则会得到：

```
x=A\b
Warning: Rank deficient, rank = 2  tol =   8.6569e-015.
x =  0.8333
       0
    0.1667
```

此解是方程Ax=b在最小二乘意义下的优化解，求解过程是先将矩阵A进行qr分解，生成矩阵Q和R，然后分两步计算x，即：y=Q'b和x=R\y。可以验证，Ax在圆整误差范围内是等于b的，方程有无数个解，x只是其中的一个。返回的结果中，tol用来决定矩阵R的对角线元素是否可以被忽略，tol的值为max(size(A))*eps*abs(R(1，1))。

5.5.5 矩阵的奇异值分解

矩阵的奇异值分解用函数svd()来求解。

s = svd(X)　得到矩阵X的奇异值组成的向量。

[U，S，V] = svd(X)　得到一个与X具有相同维数的矩阵S，其对角线元素为递减的非负值；同时得到酉矩阵U和V，使得X=USV'。

例 34

```
 [u,s,v]=svd(A)
u = 0.1248  -0.2363     -0.9009     -0.3419
    0.2497  -0.4726 -0.1566 0.8305
    0.3745  -0.7089     0.4047      -0.4397
    0.8842  0.4671      0.0000      0
s = 13.2876      0          0
    0            2.1072     0
    0            0          0.0000
    0            0          0
v = 0.3312       -0.9049    0.2673
```

```
        0.5973      -0.0182     -0.8018
        0.7304      0.4252      0.5345
```

奇异值分解也是矩阵求秩运算的基础，对矩阵 A 进行奇异值分解 s=svd(A)，得到的向量 s 的非零元素的个数就是矩阵 A 的秩。

例 35

```
s=svd(A)
s = 13.2876
     2.1072
         0.0000
```

可见矩阵的秩为 2，用求秩运算 rank(A)可以验证这一结果。

5.5.6 矩阵的 Hessenberg 分解

用函数 H=hess()可以求矩阵的 Hessenberg 形式 H，H 的第 1 子对角线以下的元素为零元素。如果矩阵 A 是对称的或是 Hermitian 矩阵，则 H 是对角三角阵，即 diag(A,-1)非零的上三角阵。A 必须为方阵。

[P，H] = hess(A)　得到矩阵的 Hessenberg 形式 H 和一个酉矩阵 P，使得 A=P*H*P' 和 P'*P=eye(size(A))。

例 36

```
A =[ 0.4447      0.9218     0.4057
    0.6154      0.7382     0.9355
     0.7919     0.1763     0.9169]
[p,h]=hess(A)
p =    1.0000   0           0
        0       -0.6136     -0.7896
        0       -0.7896     0.6136
h = 0.4447      -0.8860     -0.4789
     -1.0030    1.3883      -0.3289
     0          0.4303      0.2668
```

5.5.7 矩阵的 schur 分解

矩阵的复 schur 形式是特征值在对角线上的上三角阵；赋值的实 schur 形式是实特征值在对角线上，而复特征值以 2×2 的块矩阵排列在对角线上。用函数 T=schur(A)求矩阵 A 的 schur 形式时，如果 A 为实矩阵，函数返回 A 的实 schur 形式，反之则返回 A 的负 schur 形式，矩阵 A 必须为方阵。用函数 rsf2csf()可以将实形式转换为复形式。

T = schur(A)　此函数得到矩阵的 schur 矩阵 T。

[U，T] = schur(A)　此函数得到矩阵的 schur 矩阵 T 和酉矩阵 U，使得 A=U*T*U'，U'*U=eye(size(A))。

例 37

```
A = [ 5    -8     6
      1     0     0
```

```
        0    1    0]
[u,t]=schur(A)
u =-0.9435   -0.2568    0.2097
   -0.3145    0.4932   -0.8111
   -0.1048    0.8311    0.5461
t =  3.0000    0.2582  -10.1829
        0    0.3457   -3.3471
        0    0.4267   ·1.6543
[U,T]=rsf2csf(u,t)
U =-0.9435           0.2029 - 0.2023i  -0.0218 - 0.1652i
  -0.3145          -0.5271 + 0.3887i   0.2524 + 0.6392i
  -0.1048          -0.2450 + 0.6550i  -0.5611 - 0.4304i
T = 3.0000           -3.5571 + 0.2034i   5.1641 + 8.0247i
      0            1.0000 + 1.0000i   0.9135 - 3.0671i
      0               0            1.0000 - 1.0000i
```

5.6 应　用

利用 MatLab 强大的矩阵的计算功能，可以求解矩阵的特征值和特征向量，对矩阵进行对角化，进而可以求解许多实际问题。

5.6.1　利用矩阵乘法求解递归问题

对于递归问题，可以构造线性递归方程组，进而构造两个向量之间的递归关系，并利用 MatLab 的矩阵计算功能来求解。如对于递归问题 u(i+1)=a1u(i)+a2u(i-1)+…+amu(i-m+1)，可以构造一个矩阵 A 及向量序列 U(k)，使得 U(i+1)=A*U(i)，其中 U(i)=[u(i+m-1) u(i+m-2)… u(i)]'，而 A 为[a1　a2　a3　…　am

```
      1   0   0   …   0
      0   1   0   …   0
      ⋮   ⋮   ⋮   ⋮   ⋮
      0   0   …   1   0]
```

即 U(k)是 m 维列向量，而 A 是 m×m 维矩阵，A 的第 1 行元素为 [a1　a2　a3　…　am]。A 的第-1 对角线元素为 1，其余元素为 0。 则 $U(k)=A\cdot U(k-1)=\cdots=A^{k-1}U(1)$，再利用矩阵乘法求得 U(k)，即可以求得 u(i)。

例 38　1202 年，Fibonacci 在一本书中提出了一个问题：如果一对兔子出生一个月后开始繁殖，每个月产生一对后代，现在有一对新生的兔子，如果兔子没有死亡，那么第 20 个月月初会有多少兔子。这就是著名的 Fibonacci 序列。

如果这对兔子出生的月份计为 0 月，这时只有一对兔子，一个月后还是一对。2 月初它们生了一对兔子，此时共有 2 对兔子。3 月初第 1 对兔子又生了一对，因此共有 3 对兔子。依次类推，设每月月初的兔子数为 u(i)，序列为 1，1，2，3，5，8，13，21，…，由以上分析可知 u(i)=u(i-1)+u(i-2)，这就是递推公式，设 U(i)=[u(i+1) u(i)]'，A=[1 1 ;1 0]

将上述递推公式用矩阵形式表示为 U(i)=AU(i-1)，其中 U(0)=[1 1]'。则可以得到 U(i)=AU(i-1)=A^2U(i-2)= …=A^iU(0)。

如果要求第 20 月月初的兔子数，求 U(20)或 U(19)即可。

在 MatLab 的工作空间中输入：

```
A = [1     1
         1     0]
u0=[1   1]'
A^19*u0
ans =  10946
        6765
A^20*u0
ans =  17711
       10946
```

可见，第 20 个月月初的兔子数为 10946 对。

5.6.2 利用矩阵对角化求解振动问题

在航空航天、力学领域都会碰到许多关于振动的问题，而绝大多数都是高节的振动问题，列出的振动方程是一组微分方程。如果对这些微分方程组进行化简，变成单变量的高阶常微分方程，求解会十分麻烦，而利用矩阵的对角化，则可以很容易地求出多维振动方程的解。如对于下列振动方程组：

```
x1"=a11*x1+a12*x2+…+a1n*xn
x2"=a21*x1+a22*x2+…+a2n*xn
 ⋮   ⋮        ⋮    ⋮    ⋮
xn"=an1*x1+an2*x2+…+ann*xn
设 X=[x1   x2   x3   …   xn]',
A=[ a11  a12  a13  …  a1n
    a21  a22  a23  …  a2n
     ⋮    ⋮    ⋮    ⋮    ⋮
    an1  an2  an3  …  ann]
```

则上述方程组变为 X"=AX，如果矩阵 A 可以对角化，即存在可逆矩阵 P，使得 inv(P)AP=diag(e1，e2， …，en)=B，令 X=PY，其中 Y=[y1，y2， …，yn]'，代入 X"=AX 可以得到 PY"=PBinv(P)PY，即 Y"=BY，此方程可以写为：

```
y1"=e1*y1
y2"=e2*y2
 ⋮  ⋮  ⋮
yn"=en*yn
```

其中每个方程都是简单的一元二阶常微分方程，可以直接求解。求出方程组的解 Y，代入 X=PY 即可以求出 X，即为原常微分方程组的解。

例 39 有 4 根完全相同的弹簧，连接 3 个完全相同的物体，弹簧的两端固定，整个装置放在光滑的桌面上，并处于自由状态。弹簧的弹性系数为 k，物体的质量为 m，设 3 个

物体离开其平衡位置的位移分别为 $x_1(t)$，　$x_2(t)$，$x_3(t)$



则振动的方程组为：

$$\begin{cases} mx_1''=-k x_1+k(x_2-x_1)=k(-2 x_1+x_2) \\ mx_2''=-k(x_2-x_1)+k(x_3-x_2)=k(x_1-2 x_2+x_3) \\ mx_3''=-k(x_3-x_2)-kx_3=k(x_2-2x_3) \end{cases}$$

方程组的初始条件为：x1(0)=x2(0)=x3(0)=0

x1'(0)=x2'(0)=x3'(0)=1m/s

在不影响解题的基本原理的前提下，为了简化计算，令 k/m=1，同时令 X=[x1　x2　x3]'，令矩阵 A=[-2　1　0]

1　-2　1

0　1　-2]

则上面的振动方程组可以简化为 X"=AX。此方程的初始条件为 X(0)=[0　0　0]'，X'(0)=[1　1　1]'。下面求矩阵 P，使得 inv(P)AP 化为对角阵:

```
eig(A)                                  %求矩阵 A 的特征值
ans =
   -0.5858
   -2.0000
   -3.4142
```

矩阵 A 有 3 个不同的特征值，说明 A 可以对角化：

```
 [P,e]=eig(A)                           %求对角化矩阵 P
P =
    0.5000   -0.7071   -0.5000
    0.7071    0.0000    0.7071
    0.5000    0.7071   -0.5000
e =
   -0.5858         0         0
         0   -2.0000         0
         0         0   -3.4142
```

可以证明，inv(P)AP=e。

求解方程 Y"=eY，展开为：

$y1''=-0.5858y_1$

$y2''=-2y_2$

$y3''=-3.4142\,y_3$

方程的初始条件变为：Y(0)=inv(P)X(0)=0，Y'(0)=inv(P)X'(0)

通过矩阵运算，得：

```
Y0d=inv(P)*[1 1 1 ]'
```

```
Y0d = 1.7071
      0
      -0.2929
即 y1(0)=0,y1'(0)=1.7071
   y2(0)=0,y2'(0)=0
   y3(0)=0,y3'(0)=-0.2929
```

求解可得：$y_1=2.2304\sin(0.7654t)$

$y_2=0$

$y_3=-0.1505\sin(1.8478t)$

由 X=P*Y 可得：

$x1=1.1152\sin(0.7654t)+0.0793\sin(1.8478t)$

$x2=1.5772\sin(0.7654t)-0.1121\sin(1.8478t)$

$x3=1.1152\sin(0.7654t)+0.0793\sin(1.8478t)$

对于高阶常微分方程，同样可以化为上述形式，解法也基本相同。

5.6.3 求解二次型的标准形

对于线性空间上的对称双线性函数，其定义其可以写为：

$Q(x_1,x_2,\cdots,x_n)=a_{11}x_1^2+a_{12}x_1x_2+\cdots+$

$a_{1n}x_1x_n+a_{21}x_2x_1+a_{22}x_2^2+\cdots+a_{2n}x_2x_n$

$+\cdots+$

$a_{n1}x_nx_1+a_{n2}x_nx_2+\cdots+a_{nn}x_n^2$

其中 aij=aji，都可以化为标准形式。设由 aij 按上面的格式组成的矩阵为 A，则 Q 可以写成 Q=X'AX，其中 $X=[x_1 \quad x_2 \quad x_3 \cdots x_n]'$，要将二次型化为标准形式 $l_1y_1^2+l_2y_2^2+\cdots+l_ny_n^2$，就是要找到一个正交矩阵 P，使得 P'AP=diag(l1，l2，l3，…，ln)。令 X=PY，则 $Q=X'AX=(PY)'AX=Y'P'APY=Y'\mathrm{diag}(l_1, l_2, \cdots, l_n)Y=l_1y_1^2+l_2y_2^2+\cdots+l_ny_n^2$，便得到化简。

例 40 化简二次型 $x_1^2+x_2^2+x_3^2+4x_1x_2+4x_1*x_3+4*x_2x_3$

```
A =[ 1    2    2
     2    1    2
     2    2    1]
[P,e]=eig(A)
P =-0.7850   0.2246   0.5774
    0.5870   0.5676   0.5774
    0.1980  -0.7921   0.5774
e = -1.0000     0          0
     0         -1.0000     0
     0          0          5.0000
rank(P) = 3
```

由 5.2 节可知，P 就是所求的正交矩阵，使得 P'AP=e，所以令 X=PY，化简后的二次型为 $Q=-y_1^2-y_2^2+5y_3^2$。

5.7 习 题

(1) 求下列矩阵的特征值和特征向量

① [1 0 0
1 −1 0
2 3 2]

② [2 1 1
0 1 −1
0 1 3]

③ [1 1 1 1
1 1 −1 −1
1 −1 1 −1
1 −1 −1 1]

(2) 求下列矩阵的特征多项式

① [2 1 1
1 0 2
3 −1 2]

② [3 −1 1
2 0 1
1 −1 2]

(3) 判断下列矩阵是否可以对角化，如果可以对角化，并求出可逆矩阵和对角阵

① [1 1 −2
4 0 4
1 −1 4]

② [1 2 3
0 1 0
2 1 2]

③[1 4
1 −2]

④ [1 0
−2 1]

(4) 将下列对角阵进行 QRQ 对角化

① [0 −1 −1
−1 0 −1
−1 −1 0]

② [3 0 1
0 2 0
1 0 3]

③ [0 0 4 1
0 0 1 4
4 1 0 0
1 4 0 0]

(5) 求下列矩阵的 Jordan 标准形

① [3 0 8
3 −1 6
−2 0 −5]

② [−4 2 10
−4 3 7
−3 1 7]

(6) 设 $f(x)=x^5-x^4+2x^3+x^2+3$，$g(x)=x^4+x^2+x-1$，求 f(x)和 g(x)的最大公因式和最小公倍式。

(7) 求下列多项式的根

① $x^3-6x^2+15x-14$

② $x^4-5x^3-14x^2-10x-3$

(8) 对下列对称正定阵进行 llt 分解、ldl 分解、lu 及 qr 分解

① [2 1 1
1 2 1
1 1 2]

② [3 −1 0
−1 3 1
0 1 1]

(9) 求正交线性变换 X=PY，化下列实二次形为标准形

① $2x_1x_2+2x_1x_3+2x_2x_3$

② $3x_1^2+3x_2^2+4x_1x_2+8x_1x_3+4x_2x_3$

(10) 在求关于振动的例题中，如果振子的个数增加为 4 个，求各个振子的振动方程

第 6 章　MatLab 在微积分中的应用

学习了 MatLab 在高等代数中的应用，现在来看看 MatLab 在高等数学中的作用。高等数学最基本的概念集中在极限、导数、积分、级数等几个部分，在实际运算中这些部分也是最多。本章主要介绍 MatLab 在这几方面的应用，它们是能否顺利熟练地使用 MatLab 解决实际问题的关键。

6.1　极限、导数与微分

6.1.1　极限

从高等数学的发展来看，极限概念占有非常重要的位置。但在实际运算中，由于需要很多技巧，因而还是比较复杂。而 MatLab 提供的求极限命令 limit 则可以完成运算。其使用格式为：

```
limit(expression, var)
```

该格式将对符号表达式中的变量 var 进行其趋近于 0 时的求极限运算。

例 1

```
syms x y a
f=sin(x+2*y)
limit(f, y)
```

结果为

```
ans=sin(x)
```

当然，如果对系统的默认变量求极限时，也可不说明变量名。

```
limit(f)
```

结果为：

```
sin(2*y)
```

当需要求变量 var 在趋近于 a 时的值时，可用如下表达式：

```
limit(expression, var, a)
```

例 2

```
dsin=(sin(x+y)-sin(y))/y;
limit(dsin, x, a)
```

结果为

```
ans=cos(a)
```

6.1.2 导数与微分

有了极限的概念，理解导数与微分的概念就容易多了。先讲导数，简单地说，函数 f(x，y，z，……)在某一点(x_0，y_0，z_0，……)的增长率即为此函数在该点的导数。对一元函数来说，严格定义如下：

$$f'(x)=\lim_{\Delta x\to 0}\frac{f(x+\Delta x)-f(x)}{\Delta x}$$

可以用前面讲的 limit 命令来求各种函数的导数，但利用导数的基本概念，可以轻松地进行计算。

在 MatLab 中，求函数的导数及其他一些类似运算均可由 diff 命令来完成。其具体使用方法如下：

1. 函数 f(x)=log(x) (即 lgx)的求导

格式为：

```
diff(f)
```

例 3

```
syms x                    %或 syms x
f=log(x3);                %或diff(log(x))
diff(f)
```

结果为：

```
ans=1/x
```

例 4 求 $f(x)=(x+e^x\sin x)^{1/2}$ 的导数

```
syms x
f=(x+exp(x)*sin(x))^(1/2)
diff(f)
```

解得：

```
ans=1/2/(x+exp(x)*sin(x))^(1/2)*(1+exp(x)*sin(x)+exp(x)*cos(x))
```

用 pretty 来化简一下：

```
pretty(ans)
                  1 + exp(x) sin(x) + exp(x) cos(x)
                  1/2 -----------------------------------
                                    1/2
```

```
                    (x + exp(x) sin(x))
```

这结果显得整齐易懂。

2. 用diff求函数的高阶导数

高数里求 $y=e^{-2x}\cos(3x^{1/2})$的 4 阶导数很复杂。用 MatLab，只要在 diff(function)中的 function 后加入想要的阶数，就能得到结果。不过如果阶数太高，可能导致死机。其使用格式为：

```
diff(f, n)
```

例 5

```
y= exp(-2*x)*cos(3*(x)^(1/2));
diff(y, 4)
ans=16*exp(-2*x)*cos(3*x^(1/2))+48*exp(-2*x)*sin(3*x^(1/2))/x^(1/2)
-54*exp(-2*x)*cos(3*x^(1/2))/x-9*exp(-2*x)*sin(3*x^(1/2))/x^(3/2)-351/1
6*exp(-2*x)*cos(3*x^(1/2))/x^2-9/8*exp(-2*x)*sin(3*x^(1/2))/x^(5/2)-135
/16*exp(-2*x)*cos(3*x^(1/2))/x^3+45/16*exp(-2*x)*sin(3*x^(1/2))/x^(7/2)

pretty(ans)

pretty(ans)

                                                     1/2                            1/2
                           1/2     exp(-2 x) sin(3 x   )        exp(-2 x) cos(3 x   )
16 exp(-2 x) cos(3 x   ) + 48 ---------------------- - 54 ----------------------
                                        1/2                             x
                                       x

                             1/2                                  1/2
       exp(-2 x) sin(3 x   )     351 exp(-2 x) cos(3 x   )
   - 9 ----------------------  - --- ----------------------
                 3/2             16              2
                x                               x

                               1/2                                  1/2
       exp(-2 x) sin(3 x   )     135 exp(-2 x) cos(3 x   )
   - 9/8 ----------------------  - --- ----------------------
                 5/2             16              3
                x                               x

                             1/2
       45 exp(-2 x) sin(3 x   )
     + -- ----------------------
       16           7/2
                   x
```

3. 多元函数的求导

与求极限相同，只要在 diff 命令中加入对所求变量的说明就可以了。格式为：

```
diff(function, 'variable', n)
```

其中 n 为求导阶数。

例 6

```
syms x y z
f=x*sin(exp(y^(1/2)))/z;
diff(f, y)
```

解得：

```
ans = 1/2*x*cos(exp(1/2*y))*exp(1/2*y)/z

pretty(ans)

          x cos(exp(1/2 y)) exp(1/2 y)
      1/2 ------------------------------
                       z
```

4. **对抽象函数的求导**

对抽象函数的求导，是 MatLab 十分特别的功能之一。它的操作十分简单，与其他函数求导的步骤一样，先说明函数的自变量，再说明函数的形式，最后用 diff 求导。同时与其他求导结果一样，也可以用 pretty 函数得到一个符合日常书写习惯的表达式。

例 7

```
syms x y z a b c
f=sym('f(x, y, z)');
g=sin(a*x+y^b+c^z);
pretty(diff(f))

                 d
                 -- f(x,  y,  z)
                 dx
pretty(diff(f, z))

                 d
                 -- f(x,  y,  z)
                 dz
pretty(diff(f, y, 2))

                  2
                  d
                  --- f(x,  y,  z)
                  2
                  dy
pretty(diff(diff(f, y, 2)), 'z')

                   3
                   d
                   ------ f(x,  y,  z)
                   2
```

```
                                dy  dx
pretty(diff(g, z))

                                b    z   z
                                cos(a x + y  + c ) c  log(c)
```

6.2 积　　分

积分与微分、求导一样是高等数学中最基本的运算之一。它的运算思路很简单，就是求一条曲线、一个空间曲面及空间曲体在一定坐标系下对应的面积或体积。在 MatLab 中，int 是求数值和符号积分的基本命令，它的功能十分强大，是计算中使用很多的工具。

6.2.1 不定积分

不定积分是积分运算中最基本的运算，同求导一样，它的步骤繁多且容易出错。int 命令很好地解决了这个问题，只要写出待积分的函数，int 就可很快求出积分函数。其使用格式为：

```
int(f)
int(f, var)
```

例 8

```
syms x y z;
int(sin(x*y+z))
ans= -cos(x*y+z)/y
```

如果对 z 积分，则应在 int 命令后说明：

```
int(sin(x*y+z), z)
ans=-cos(x*y+z)
```

值得一提的是，在目前广泛使用的数学软件中，MatLab 的符号积分命令 int 是各家公司同类软件相同命令中数一数二的。有很多函数别的软件积不出来，而 int 则可以很快得出结果，如函数：

```
F(x)=e-2x
```

6.2.2 定积分及广义积分

在 MatLab 中只要在 int 命令中加入积分限，就可求得函数在积分上下限间的积分值。格式如下：

int(function，var，积分下限，积分上限)

例 9

```
syms x y
ansa=int(cos(x), 0, pi/6);
```

```
ansb=int(x^y, y, 0, pi/6);
```

解得:

```
ansa=1/2
ansb=1/log(x)*(x^pi)^(1/6)-1/log(x)
```

由 ansa=1/2 可知，int 命令解得的值为精确解。

当积分限由某一具体数值变为正负无穷时，定积分便转变为广义积分，也只需将积分限变为无穷，就可得到相应函数的广义积分值。现举例说明。

例 10 求函数 $f(x)=1/x$，$g(x)=1(1+x^2)$在 1 到正无穷的积分。

```
syms x
f=1/x;
g=1/(1+x^2);
intf=int(f, 1, inf);
intg=int(g, 1, inf);
```

解得:

```
intf=inf
intg=1/4*pi
```

例 11

求函数 $f(x)= 1/(x^2+2x+3)$，$g(x)=1/(x^2+2x-3)$在负无穷到正无穷的积分。

```
syms x
f=1/(x^2+2*x+3);
g=1/(x^2+2*x-3);
intf=int(f, -inf, inf);
intg=int(g, -inf, inf);
```

解得:

```
intf=1/2*pi*2^(1/2)
Warning: Explicit integral could not be found.
 In E:\toolbox\symbolic\@sym\int.m at line 58
intg=int(1/(x^2+2*x-3), x = -inf .. inf)
```

结果说明 f(x)在整个数轴上可积，而 g(x)在整个数轴上不可积。为什么呢？看了两个函数的图像就明白了。

```
ezplot(f, -10, 10);
ezplot(g, -10, 10);
```

结果如图 3.1，3.2 所示。

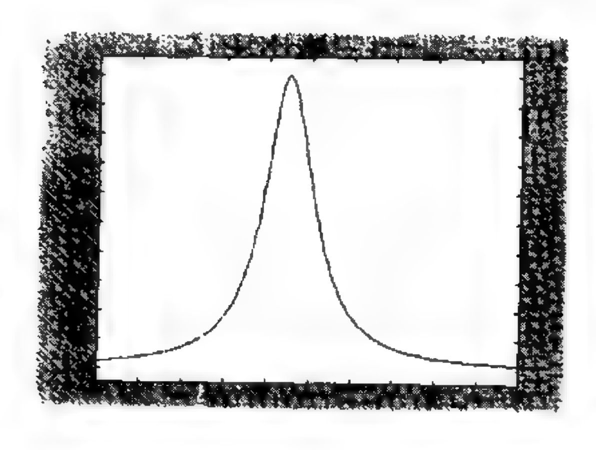


图 3.1　f(x)的函数图像

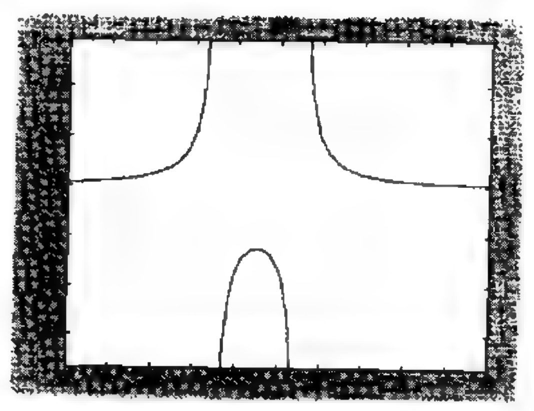


图 3.2　g(x)的函数图像

原来，g(x)在数轴上有不可积的奇点。

6.3　化简、提取与替换代入

6.3.1　化简

不管是用 MatLab 进行计算，还是用手在纸上推算，都会出现要将前面已知式子代入新式和对初步运算结果进行化简的情况。如待算式子太复杂，不论是化简还是代入都容易出错。现在，介绍一下 MatLab 中的一个非常有特点的功能块：化简与代入。

1. pretty 命令

在说明 MatLab 的化简与带入功能前，首先介绍一下 MatLab 中的将代数式化为手写格式的格式转化命令 pretty。

在 MatLab 的使用中，MatLab 的功能虽强，但它的计算结果却并不直观，特别是乘和幂次运算，*和^在式子中使人看着觉得繁琐，而 pretty 命令则解决了这个问题。它的用法很简单。格式如下：

如 A 为待转化格式的代数式，命令 pretty(A)即可将 A 由机器格式转化为手写格式，而且在转化过程中不会对 A 式进行任何化简或展开。

例 12　试将 $f=(x+y)(a+b^c)^z/(x+a)^2$ 和 $g=(a+b^c)^z/x(x+a)^2*x+(a+b^c)^z/y(x+a)^2$ 用 pretty 命令转化为手写格式，并判断两式是否相等。

实际操作如下：

```
syms x y z a b c
f=(x+y)*(a+b^c)^z/(x+a)^2;
g=(a+b^c)^z/(x+a)^2*x+(a+b^c)^z/(x+a)^2*y;
pretty(f)
pretty(g)
```

经转化，结果为：

```
pretty(f)

                 c z
       (x + y) (a + b )
       ----------------
                  2
           (x + a)
pretty(g)

           c z         c z
      (a + b ) x  (a + b ) y
      ---------- + ----------
             2           2
       (x + a)     (x + a)
```

这两个式子看似不同，下面让两式相减。

```
f-g
ans=(x+y)*(a+b^c)^z/(x+a)^2-(a+b^c)^z/(x+a)^2*x-(a+b^c)^z/(x+a)^2*y
```

此次相减结果并不为 0，再用 pretty 命令观察一下：

```
pretty(ans)

              c z         c z         c z
    (x + y) (a + b )  (a + b ) x  (a + b ) y
    ---------------- - ---------- - ----------
               2             2           2
        (x + a)        (x + a)     (x + a)
```

由上可见，f-g 的结果只是两式在形式上相减了一下，而完全没有进行化简。两式是否最终相等呢？MatLab 中的化简功能正好可对其进行检验。

2. MatLab 的化简命令

与纸上的化简过程一样，MatLab 的化简命令有多种，分别对应纸上的不同方法。它们分别是：降幂排列法(collect)，展开法(expand)，重叠法(horner)，因式分解法(factor)，单一化简(simplify)，不定化简(simple)等 6 种方法。下面一一说明。

- 降幂排列法(collect)

降幂排列法是各种化简方法中最简单的一种，在 MatLab 中由 collect 命令完成。它的用法简单，格式为：

```
collect(A)
```

如要对非默认变量进行降幂排列，则要声明该变量名，格式为：

```
collect(A, name_of_varible)
```

现举一例说明其具体使用方法及过程。

例 13　化简以下两式：

(1) $t=(ax+3bx^4-3)^2+24a(cx^7+ax^3+4cx^2-8b^1)^2-67a((2+7x)^5-34ax+4c)$按 x 降幂排列

(2) $tt=t+e^{3x}a^{-2}+e^{3x}x^{20}$ 按 a 降幂排列

运算过程为：

```
    syms x a b c
    t=(a*x+3*b*x^4-3)^2+24*a*(c*x^7+a*x^3+c*4*x^2-b*8*x^(-1))^2-67*a*((
2+7*x)^5-34*a*x+c*4);
    tt=t+exp(3*x)*a^(-2)+exp(3*x)*x^20*a;
    anst=collect(t);
    anstt=collect(tt, a);
```

结果为：

```
    anst=24*a*c^2*x^14+48*a^2*c*x^10+192*a*c^2*x^9+9*b^2*x^8+24*a*(-16*
b*c+a^2)*x^6+(192*a^2*c+6*a*b-1126069*a)*x^5+(384*a*c^2-18*b-1608670*a)
*x^4-919240*a*x^3+(-262640*a-384*a^2*b+a^2)*x^2+(-6*a-67*a*(560-34*a)-1
536*a*b*c)*x+9-67*a*(32+4*c)+1536*a*b^2/x^2

    anstt=24*a^3*x^6+(x^2+2278*x+48*(c*x^7+4*c*x^2-8*b/x)*x^3)*a^2+(2*(
3*b*x^4-3)*x-67*(2+7*x)^5-268*c+24*(c*x^7+4*c*x^2-8*b/x)^2+exp(3*x)*x^2
0)*a+(3*b*x^4-3)^2+exp(3*x)/a^2
```

- 展开法(expand)

展开法即是将代数式中所有的括号打开，将变量解放出来，但得出的结果并不进行任何整理和幂次排列，只将其凌乱的堆在一起。现在，用展开法化简上例中的 t。

例 14

```
    expand(t)
    ans=a^2*x^2+6*a*x^5*b-37526*a*x+9*b^2*x^8-18*b*x^4+9+24*a*c^2*x^14+
48*a^2*c*x^10+192*a*c^2*x^9-384*a*c*x^6*b+24*a^3*x^6+192*a^2*x^5*c-384*
a^2*x^2*b+384*a*c^2*x^4-1536*a*c*x*b+1536*a*b^2/x^2-2144*a-262640*a*x^2
-919240*a*x^3-1608670*a*x^4-1126069*a*x^5+2278*a^2*x-268*c*a
```

上式将 a，b，c 也展出了括号，比起用幂次排列法化简的结果长了许多。

- 重叠法(horner)

重叠法是一种很特别的代数式的整理化简方法。它的化简方法是将代数式尽量化为 ax(bx(cx(...(zx+z')+y')+...)+b')+a'的形式。在 MatLab 中，将代数式 A 以重叠法化简的命令 horner 使用起来同样简单，格式为：

```
    horner(A)
```

下面举两个化简的实例。

例 15 用重叠化简法化简下面两式

(1) $m=x^5y^3+(xy^2+6)^2+43y$

(2) $n=x^3-5x^2+11x-13$

具体操作为：

```
    syms x y
    m=x^5*y^3+(6+x*y^2)^2+43*y;
```

```
n=x^3-5*x^2+11*x-13;
ansn=horner(n);
ansm=horner(m);
```

结果为：

```
ansn=-13+(11+(-5+x)*x)*x
ansm=43*y+36+(12*y^2+(y^4+y^3*x^3)*x)*x
```

- 因式分解法(factor)

因式分解法是化简方法中最常用的一种方法，它的目的就是将代数式 A 化为由 x 的一次项为单位的连乘积的形式。它在 MatLab 中的命令名为 factor。使用格式为：

```
factor(A)
```

例 16

(1) $f=16741x^6+3375x^9-12825x^8+9495x^7+1376x^2-512-2176x-22030x^5-4780x^4+11336x^3$

(2) $g=x^{11}-x^9-12x^7+16x^5+3x^{10}-11x^8-4x^6+48x^4+32x^3-32x^2-48x-16$

进行因式分解。

具体操作为：

```
syms x
f=16741*x^6+3375*x^9-12825*x^8+9495*x^7+1376*x^2-512-2176*x-22030*x
^5-4780*x^4+11336*x^3;
g=x^11-x^9-12*x^7+16*x^5+3*x^10-11*x^8-4*x^6+48*x^4+32*x^3-32*x^2-4
8*x-16;
ansf=factor(f);
ansg=factor(g);
```

结果为：

```
ansf=(x+1)*(x-1)^2*(5*x+2)^3*(3*x-4)^3
ansg=(x^2+2)*(x+1)^3*(x^2-2)^3
```

- 一般化简(simplify)

在 MatLab 中，一般化简是指代数式在考虑了求和、积分、平方运算法则，三角函数、指数函数、对数函数、Bessel 函数、hypergeometric 函数、gamma 函数的运算性质，经计算机比较后转化的一种认为相对简单的形式。此种转化只列出结果，用户并不知道这种形式是经何种变换后得到的。但在普通的化简运算中，一般化简法(simplify)倒不失为一种简便快捷的化简方法。

它的使用方法同前，格式为：

```
simplify(A)
```

例 17　试将下列式子用一般化简法进行化简

(1) $a=x(x(x-8)+11)+6$

(2) $b=\log(xy)$

(3) $c=e^x e^y e^z$

(4) d=besselj(2, x)-2besselj(1, x)/x+ besselj(0, x)

(5) $e=\sin^2x+\cos^2x$

具体操作为：

```
syms x y z
a=x*(x*(x-8)+11)+6;
b=log(x*y);
c=exp(x)*exp(y)*exp(z);
d=besselj(2, x)-2*besselj(1, x)/x+besselj(0, x);
e=cos(x)^2+sin(x)^2;
ansa=simplify(a);
ansb=simplify(b);
ansc=simplify(c);
ansd=simplify(d);
anse=simplify(e);
```

结果为：

```
ansa=x^3-8*x^2+11*x+6
ansb=log(x*y)
ansc=exp(x+y+z)
ansd=0
anse=1
```

可见，一般化简法的功能较前几种更广泛一些，但目的性没有前几种那么明确。在前面介绍 pretty 命令时留下的那个小问题，f 到底等不等于 g 呢，用 simplify 命令化简一下，自然会得到结果。而且这个结果再次说明，命令 pretty 对代数式只是转化格式，并未进行任何的化简。

- 不定化简(simple)

不定化简命令 simple 综合了前面几种化简方法的优点，但也略显笨拙。因为它不仅自动将前面每一种化简方法都试了一遍，还尝试了 4、5 种转化方法，最后还一一将这些结果列了出来。列出的结果往往多得超出 3、4 屏，用户可细细观察挑选。

下面介绍一下 simple 命令中的几个转化运算：

```
combine(trig)
```

以三角函数的运算性质为主对代数式进行化简。

```
convert(exp)
```

将代数式尽量转化为由 ex、eix 表示的指数代数式。

```
convert(sincos)
```

将代数式尽量转化为由 sin(x)、cos(x)表示的式子。

```
convert(tan)
```

将代数式尽量转化为由 tan(x)表示的式子。

现在，看看 simple 命令的效果。simple 同前几个命令的使用格式相同，为：

```
simple(A)
```

例 18 试将以下两式用不定化简法化简

(1) $g=\sin^2x+\cos^2x$

(2) $f=(x^2+2x+1)/(x+1)+x\cos x\mathrm{tg}x$

操作如下：

```
syms x
f=(x^2+2*x+1)/(x+1)+cos(x)*x*tan(x);
g=sin(x)^2+cos(x)^2;
simple(g)
simple(f)
```

结果分别为：

```
g:
simplify:
1
radsimp:
sin(x)^2+cos(x)^2
combine(trig):
1
factor:
sin(x)^2+cos(x)^2
expand:
sin(x)^2+cos(x)^2
convert(exp):
-1/4*(exp(i*x)-1/exp(i*x))^2+(1/2*exp(i*x)+1/2/exp(i*x))^2
convert(sincos):
sin(x)^2+cos(x)^2
convert(tan):
4*tan(1/2*x)^2/(1+tan(1/2*x)^2)^2+(1-tan(1/2*x)^2)^2/(1+tan(1/2*x)^
2)^2
collect(x):
sin(x)^2+cos(x)^2
ans =1

f:
<
simplify:
x+1+x*sin(x)
radsimp:
x+1+cos(x)*x*tan(x)
combine(trig):
x+1+cos(x)*x*tan(x)
factor:
```

```
x+1+cos(x)*x*tan(x)
expand:
1/(x+1)*x^2+2/(x+1)*x+1/(x+1)+cos(x)*x*tan(x)
convert(exp):
(x^2+2*x+1)/(x+1)-i*(1/2*exp(i*x)+1/2/exp(i*x))*x*(exp(i*x)^2-1)/(e
xp(i*x)^2+1)
convert(sincos):
(x^2+2*x+1)/(x+1)+x*sin(x)
convert(tan):
(x^2+2*x+1)/(x+1)+(1-tan(1/2*x)^2)/(1+tan(1/2*x)^2)*x*tan(x)
collect(x):
(x^2+2*x+1)/(x+1)+cos(x)*x*tan(x)
ans =
x+1+x*sin(x)

<
```

以上是 MatLab 的主要化简功能，下面进一步学习代入功能，当对此两者均有了一定的把握之后，就表明向数学运算的无纸化迈进了一步。

6.3.2　提取与替换代入

在 MatLab 中，代入命令有两个，分别是 subexpr 和 subs。这两条命令各有优势，用起来也都很方便，下面分别予以介绍。

- **提取(subexpr)**

在进行繁琐的数学运算中，经常会碰到类似这样的情况：在得到的方程的解中，有几个非常长的因子在解中出现很多遍，不管是在纸上还是在屏幕上，它不仅使式子过长变得难看，而且在转抄或粘贴时非常容易出错。MatLab 的 subexpr 命令可以解决这个问题。它能用一个语句完成筛选相同因子和整理式子的复杂工作。

在使用中，subexpr 命令可以带一个或是两个参数。它的完整使用格式为：

[Y，SIGMA] = subexpr(X，SIGMA)

或 [Y，SIGMA] = subexpr (X，'SIGMA')

式中各参数含义如下：

X　待整理的代数式或代数式的矩阵。

SIGMA 在整理过程中提出的各种因子将以矩阵的格式保存在名为 SIGMA 的变量中。

Y　经提取各种因子后，整理完毕的代数式或其矩阵将被保存于 Y 矩阵中。

例 19

```
t=solve('a*x^3+b*x^2+c*x+d=0')   %(将方程的解赋予变量 t)
t=
[1/6/a*(36*c*b*a-108*d*a^2-8*b^3+12*3^(1/2)*(4*c^3*a-c^2*b^2-18*c*b
*a*d+27*d^2*a^2+4*d*b^3)^(1/2)*a)^(1/3)-2/3*(3*c*a-b^2)/a/(36*c*b*a-108
*d*a^2-8*b^3+12*3^(1/2)*(4*c^3*a-c^2*b^2-18*c*b*a*d+27*d^2*a^2+4*d*b^3)
^(1/2)*a)^(1/3)-1/3*b/a]
[-1/12/a*(36*c*b*a-108*d*a^2-8*b^3+12*3^(1/2)*(4*c^3*a-c^2*b^2-18*c
```

```
*b*a*d+27*d^2*a^2+4*d*b^3)^(1/2)*a)^(1/3)+1/3*(3*c*a-b^2)/a/(36*c*b*a-1
08*d*a^2-8*b^3+12*3^(1/2)*(4*c^3*a-c^2*b^2-18*c*b*a*d+27*d^2*a^2+4*d*b^
3)^(1/2)*a)^(1/3)-1/3*b/a+1/2*i*3^(1/2)*(1/6/a*(36*c*b*a-108*d*a^2-8*b^
3+12*3^(1/2)*(4*c^3*a-c^2*b^2-18*c*b*a*d+27*d^2*a^2+4*d*b^3)^(1/2)*a)^(
1/3)+2/3*(3*c*a-b^2)/a/(36*c*b*a-108*d*a^2-8*b^3+12*3^(1/2)*(4*c^3*a-c^
2*b^2-18*c*b*a*d+27*d^2*a^2+4*d*b^3)^(1/2)*a)^(1/3))]
    [-1/12/a*(36*c*b*a-108*d*a^2-8*b^3+12*3^(1/2)*(4*c^3*a-c^2*b^2-18*c
*b*a*d+27*d^2*a^2+4*d*b^3)^(1/2)*a)^(1/3)+1/3*(3*c*a-b^2)/a/(36*c*b*a-1
08*d*a^2-8*b^3+12*3^(1/2)*(4*c^3*a-c^2*b^2-18*c*b*a*d+27*d^2*a^2+4*d*b^
3)^(1/2)*a)^(1/3)-1/3*b/a-1/2*i*3^(1/2)*(1/6/a*(36*c*b*a-108*d*a^2-8*b^
3+12*3^(1/2)*(4*c^3*a-c^2*b^2-18*c*b*a*d+27*d^2*a^2+4*d*b^3)^(1/2)*a)^(
1/3)+2/3*(3*c*a-b^2)/a/(36*c*b*a-108*d*a^2-8*b^3+12*3^(1/2)*(4*c^3*a-c^
2*b^2-18*c*b*a*d+27*d^2*a^2+4*d*b^3)^(1/2)*a)^(1/3))]

    [r, s]=subexpr(t, 's')
    r =
    [1/6/a*s^(1/3)-2/3*(3*c*a-b^2)/a/s^(1/3)-1/3*b/a]
    [-1/12/a*s^(1/3)+1/3*(3*c*a-b^2)/a/s^(1/3)1/3*b/a+1/2*i*3^(1/2)
*(1/6/a*s^(1/3)+2/3*(3*c*a-b^2)/a/s^(1/3))]
    [     -1/12/a*s^(1/3)+1/3*(3*c*a-b^2)/a/s^(1/3)1/3*b/a1/2*i*3^(1/2)*
(1/6/a*s^(1/3)+2/3*(3*c*a-b^2)/a/s^(1/3))]
    s=36*c*b*a-108*d*a^2-8*b^3+12*3^(1/2)*(4*c^3*a-c^2*b^2-18*c*b*a*d+2
7*d^2*a^2+4*d*b^3)^(1/2)*a
```

上例中，s 为一代数式，可试将 a，b，c，d 之一、二改为某一具体数字，再用 subexpr 命令，观察一下 s 的变化。

subexpr 的简化使用格式为：

```
Y=subexpr(X)
```

式中 X，Y 含义同前，此时 X 式中的相同因子将被保存在默认名为 SIGMA 的变量中。这个操作很简单，就不再举例。

- 代入(subs)

在 MatLab 中，将一代数式带入另一式中的操作命令名为 subs。它的用法比较灵活，而且适用范围较广。基本使用格式为：

```
SS=subs(S, OLD, NEW)
```

上式中各项参数的含义为：

S　代数式名。

OLD　代数式 S 中的将要被替换的旧变量名。

NEW　将要替换 OLD 的新变量或代数式。

SS　替换后的新代数式。

例 20

将 $f=ax^2+bx+c$ 中的变量 x 分别替换为 y，m+nt

```
syms x a b c y m n t
f=a*x^2+b*x+c;
ansf=subs(f, x, y);
ansff=subs(f, x, 'm+n*t');
```

结果为：

```
ansf=a*y^2+b*y+c
ansff=a*(m+n*t)^2+b*(m+n*t)+c
```

另外，在用 MatLab 中的 subs 命令时，会发现系统按 SS=subs(S，OLD，NEW)的命令格式执行，却没有结果或是错误结果。原因很可能是：MatLab 为了与以前的版本兼容，subs 命令的格式变为 SS=subs(S，NEW，OLD)。如果是这样，那就要按后面的命令格式进行计算了。

知道了 subs 命令的基本使用格式，下面介绍它的几个简化命令格式和更广泛的用途。

如果要替换的变量也是系统按独立变量规则确认的变量，则 subs 命令的使用格式可简化为： SS=subs(S，NEW)。因此，可以试试前面例子中两个 subs 命令中的参数 x 是不是均可以省略不写。

如果代数式 S 中的任意变量在用 subs 命令前已经被赋值，则不管是数值型还是字符型，命令 subs(S)都将其具体值代入相应变量，完成替换并进行相应运算。

例 21

```
syms a b c x y
f=a*b+c/x*y;
a='we';
b=1;
c=4;
x='aw';
y=5;
subs(f)
```

结果为：

```
ans=we+20/aw
```

subs 命令不但可进行单一变量的替换，还可进行多个变量的同时替换和多个矩阵的同时替换。它们的替换命令格式完全相同，只是进行替换新老变量时要分别用大括号({ })括起来。

下面的例子很好地说明了大括号的位置和用法。

例 22

```
subs(cos(a)^2+cos(b)^2, {a, b}, {'alpha', 2})
ans=cos(alpha)^2+cos(2)^2

subs(exp(x*y), 'y', -magic(3))
ans=
[ exp(-8*x),    exp(-x),  exp(-6*x)]
```

```
[ exp(-3*x), exp(-5*x), exp(-7*x)]
[ exp(-4*x), exp(-9*x), exp(-2*x)]

subs(x*y, {x, y}, {[0 5 1;6 -7 3], [2 -3 5;8 -5 1]})
ans=
 0   -15    5
48    35    3
```

现在，提取与替换代入部分就全部介绍完了。这一部分内容不多但很灵活，此处掌握不精，到后面使用时就会绕很大的弯子，甚至完不成一个在运算上很简单的题目。

6.4 级数求和

MatLab 的级数求和命令功能非常强大，symsum 是 MatLab 中符号运算工具箱(Symbolic Math Toolbox)主要的级数求和命令。由于级数求和是高等数学中非常重要的一部分，下面将详细介绍求和命令 symsum 的用法。

6.4.1 symsum(s)

s 为待求和的级数的通项表达式。

命令 symsum(s)的功能是求出 s 关于系统默认变量如 k 的由 0 到 k-1 的有限项的和。如不能确定 s 的默认变量，则可用 findsym(s)命令来查得。

具体应用请见下例。

例 23

试求 $s=ac^n$；$t=m^{2\sin(n)}$的 n 由 0 至 n-1 的和。

```
syms a m n c
s=a*c^n;
t=m^2*sin(n);
anss=symsum(s);
anst=symsum(t);
```

结果为：

```
anss=
a*c^n/(c-1)
anst=
-1/2*m^2*sin(n)+1/2*m^2*sin(1)/(cos(1)-1)*cos(n)
```

6.4.2 symsum(s, v)

v 为求和变量。求和将由 v 等于 1 求至 v-1。

当不能确定自己所需的变量是系统的默认变量，或已知其不是默认变量时，需在 symsum 命令中加入对求和变量的说明，格式为：

```
symsum(s, v)
```

例 22

```
syms a q n
s=a*q^n;
anss=symsum(s);
anst=symsum(s, n);
```

结果为：

```
anss=
sum(a*q^n, q)
anst=
a*q^n/(q-1)
```

结果 anss 说明，symsum(s)是以 q 为求和变量进行运算的，由于结果不能进行化简，所以系统给出了前面的答案。

6.4.3 symsum(s, v, a, b)

前面所介绍的，一直是由 0 到 n-1 的固定长度的级数求和。能不能任选一段进行求和或将求和一直继续到正无穷呢？可以。只要在命令后面补充上对求和起点和终点的说明就可让 MatLab 完成计算。格式为：

```
symsum(s, v, a, b)
```

具体使用请见下例。

例 25

```
syms x y z a b c m n
f=a/x;
g=b/y^2;
h=c/z^3;
i=cos(a)/(2^a);
ansf1=symsum(f, x, 1, 10);
ansf2=symsum(f, x, 1, inf);
ansg1=symsum(f, 1, 10);
ansg2=symsum(f, 1, inf);
ansh1=symsum(f, 1, 10);
ansh2=symsum(f, 1, inf);
ansi1=symsum(f, a, 1, inf);
ansi2=symsum(f, a, inf);
```

结果为：

```
ansf1=
7381/2520*a
ansf2=
signum(a)*inf+a*eulergamma
```

```
ansg1=
 1968329/1270080*b
ansg2=
 1/6*b*pi^2
ansh1=
 19164113947/16003008000*c
ansh2=
 c*zeta(3)
ansi1=
 (cos(1)-2)/(-5+4*cos(1))*cos(1)-(cos(1)^2-1)/(-5+4*cos(1))
ansi2=

2*(cos(1)-2)/(-5+4*cos(1))*cos(a)/(2^a)-2*(cos(1)^2-1)/(-5+4*cos(1))/si
n(1)/(2^a)*sin(a)
```

请详细比较上面各例的不同之处，并多试一试。

有了 symsum 命令和 abs 命令，判断一个级数是否是绝对收敛就容易多了。

6.5 泰勒、傅里叶级数展开

6.5.1 一元函数 taylor 展开

泰勒展开是高等数学中遇到的第一个级数展开，对它是否有深入的理解直接影响到对更深数学知识的学习和领悟。展开的实质，就是要将自变量 x 函数表示成 x^n(n 由 0 到无穷)的和的形式。当理解了泰勒展开的原理后，对任意函数进行泰勒展开就成了一项机械的工作，完全可以交给计算机来完成。

taylor 是 MatLab 中用来完成泰勒展开操作的命令。用它可以完成各种复杂的泰勒展开运算，以下将详细加以介绍。

1. taylor(f)

f 为待展开的函数表达式。命令 taylor(f)将求解出函数 f 关于其默认变量的麦克劳林型的 6 阶近似展开。

例 26

```
syms x y a b u v
f=a*sin(x)*y^x+u*cos(v);
g=a*exp(v)+b*v^u;
ansf=taylor(f);
ansg=taylor(g);
```

结果为：

```
ansf=
```

```
u*cos(v)+a*x+a*log(y)*x^2+(1/2*a*log(y)^2-1/6*a)*x^3+(1/6*a*log(y)^
3-1/6*a*log(y))*x^4+(1/24*a*log(y)^4-1/12*a*log(y)^2+1/120*a)*x^5
ansg=
a+b+(a+b*log(u))*v+(1/2*a+1/2*b*log(u)^2)*v^2+(1/6*b*log(u)^3+1/6*a
)*v^3+(1/24*a+1/24*b*log(u)^4)*v^4+(1/120*a+1/120*b*log(u)^5)*v^5
```

2. taylor(f, n)

taylor(f)命令只能求函数表达式 f 的 6 阶麦克劳林型泰勒展开式。如要求任意阶则要在 taylor 命令后补加求阶参数 n，这样，求函数 f 的 1000 阶的泰勒展开式也是没问题的。

例 27

```
f=sin(x)+exp(x)*tan(x);
taylor(f, 20)
ans=
2*x+x^2+2/3*x^3+1/2*x^4+7/20*x^5+71/360*x^6+19/140*x^7+19/240*x^8+1
423/25920*x^9+58141/1814400*x^10+221941/9979200*x^11+15707/1209600*x^12
+850397/94348800*x^13+229395011/43589145600*x^14+56867491/15567552000*x
^15+18973501/8895744000*x^16+37613545063/25406244864000*x^17+2767173538
681/3201186852864000*x^18+2606757319183/4344467871744000*x^19

g=exp(x);
taylor(g, 30)
ans=
1+1/40320*x^8+1/5040*x^7+x+1/2*x^2+1/362880*x^9+1/24*x^4+1/6*x^3+1/
120*x^5+1/720*x^6+1/3628800*x^10+1/39916800*x^11+1/479001600*x^12+1/622
7020800*x^13+1/87178291200*x^14+1/1307674368000*x^15+1/20922789888000*x
^16+1/355687428096000*x^17+1/6402373705728000*x^18+1/121645100408832000
*x^19+1/2432902008176640000*x^20+1/1124000727777607680000*x^22+1/258520
16738884976640000*x^23+1/620448401733239439360000*x^24+1/15511210043330
985984000000*x^25+1/403291461126605635584000000*x^26+1/1088886945041835
2160768000000*x^27+1/304888344611713860501504000000*x^28+1/884176199373
9701954543616000000*x^29+1/51090942171709440000*x^21
```

3. taylor(f, v)

由于在实际计算中，变量名无所不有，特别是对于多元函数。泰勒展开一定要说明对象，否则结果就与所需不同。因此，对函数中非系统默认的自变量或多元函数中的变量进行泰勒展开时，一定要在命令中加入对变量名的说明。taylor(f, v)就是其使用格式，结果是关于 v 的麦克劳林型的泰勒展开式。

例 28

```
f=3*x*sin(y)*tan(a)+exp(a*y);
taylor(f, a)
ans=
1+(y+3*x*sin(y))*a+1/2*y^2*a^2+(1/6*y^3+x*sin(y))*a^3+1/24*y^4*a^4+
(2/5*x*sin(y)+1/120*y^5)*a^5
```

```
taylor(f, y)
ans=
1+(a+3*x*tan(a))*y+1/2*y^2*a^2+(1/6*a^3-1/2*x*tan(a))*y^3+1/24*y^4*
a^4+(1/40*x*tan(a)+1/120*a^5)*y^5
```

如想求 f 的关于 a 的 10 阶麦克劳林型的泰勒展开式，命令则变为：

```
taylor(f, a, 10)
ans=
1+(y+3*x*sin(y))*a+1/2*y^2*a^2+(1/6*y^3+x*sin(y))*a^3+1/24*y^4*a^4+
(2/5*x*sin(y)+1/120*y^5)*a^5+1/720*y^6*a^6+(1/5040*y^7+17/105*x*sin(y))
*a^7+1/40320*y^8*a^8+(62/945*x*sin(y)+1/362880*y^9)*a^9
```

4. taylor(f, a)

在前面介绍的泰勒展开的例子中，不管是对哪个变量进行的都只是在变量等于零时的展开，这就局限了泰勒展开的范围。命令 taylor(f, a)的运算结果则是函数 f 在变量等于 a 处的泰勒展开结果。

例 29

```
syms x y a b
f=sin(x)*y+exp(x)*b;
ans1=taylor(f, 4);        %将函数 f 在 x=4 处进行泰勒展开
ans2=taylor(f, a);        %将函数 f 在 x=a 处进行泰勒展开
```

结果为：

```
ans1=
b+(y+b)*x+1/2*b*x^2+(1/6*b-1/6*y)*x^3
ans2=
sin(a)*y+exp(a)*b+(cos(a)*y+exp(a)*b)*(x-a)+(-1/2*sin(a)*y+1/2*exp(
a)*b)*(x-a)^2+(1/6*exp(a)*b-1/6*cos(a)*y)*(x-a)^3+(1/24*sin(a)*y+1/24*e
xp(a)*b)*(x-a)^4+(1/120*cos(a)*y+1/120*exp(a)*b)*(x-a)^5
```

有了对 taylor 命令的各个参数的了解，从而展开任意一个函数就容易了。另外，再给出 taylor 命令的包含全部参数的使用格式，为：

```
r = taylor(f, n, v, a)
```

例 30

```
syms x y a b c
f=x*besselj(2, b)+exp(y)*a;
ansf=taylor(f, b, 4, c);
```

结果为：

```
ansf=
x*besselj(2 ,  c)+exp(y)*a+x*(besselj(1 ,  c)*c-2*besselj(2 ,
c))/c*(b-c)+1/2*x*(besselj(0 , c)*c^2-3*besselj(1 , c)*c+6*besselj(2 ,
c))/c^2*(b-c)^2-1/6*x*(besselj(1, c)*c^3-12*besselj(1, c)*c+3*besselj(0,
```

```
c)*c^2+24*besselj(2, c))/c^3*(b-c)^3
```

6.5.2 多元函数的完全泰勒展开

在命令 taylor 中，所有操作均是对表达式的一个变量进行展开，而把其他变量当作常数。而当函数不只含有一个变量时，taylor 命令就无法处理。现在介绍一个能将 n 元函数作完全泰勒展开的命令 mtaylor。

完全展开命令 mtaylor 的使用格式为：

```
mtaylor(f, v)
mtaylor(f, v, n)
mtaylor(f, v, n, w)
```

其中参数具体含义为：

f 待完全展开的代数表达式。

v 式中变量名列表，格式为：[var1=p1，var2=p2，…varn=pn]。

根据列表中的变量名和值，泰勒展开将在点(p1，p2，…，pn)处进行。当列表中的元素 vari 只有变量名时，系统将默认其值为 0。

n 非负整数，用于设定展开阶数。

w 与变量名列表同维的正整数列表，用于设置相应变量在展开时的权重。

另外，命令 mtaylor 并不在 MatLab 的符号运算工具箱的命令列表中，它是 MAPLE 符号运算函数库中的命令。因此调用这个命令的方法不同以前，首先，要将命令 mtaylor 由 MAPLE 的函数库读入工作空间，然后将用到专用于调用 MAPLE“引擎”函数 maple。因此，在 MatLab 内使用完全泰勒展开命令 mtaylor 的格式为：

```
maple('readlib(mtaylor)')
maple('mtaylor(f, v, n, w)')
```

例 31

在(1，0，0)处对函数 $\sin(x^2+y^2/z)$进行完全泰勒展开。

```
maple('mtaylor(sin(x^2+y^2/z), [x=1, y, z], 3)')
```

结果为：

```
ans =
sin(1)+2*cos(1)*(x-1)+cos(1)*y^2/z+(-2*sin(1)+cos(1))*(x-1)^2-2*sin
(1)*y^2/z*(x-1)-1/2*sin(1)*y^4/z^2
```

6.5.3 傅里叶级数展开

傅里叶级数展开在高等数学中很重要，它在工程计算和理论计算中都起着非常重要的作用。

在 MatLab 中，目前还没有一个专门用于进行傅里叶级数展开的命令。不过，从其定义来看，完全可以利用 MatLab 符号的计算能力，自己编制如下一个命令：

```
function [a0, an, bn]=mfourier(f)
syms n x
a0=int(f, -pi, pi)/pi;
an=int(f*cos(n*x), -pi, pi)/pi;
bn=int(f*sin(n*x), -pi, pi)/pi;
```

这个函数的使用很简单，只要将待展开的函数表达式赋给一个符号变量，然后用这个变量作为命令 mfourier 的参数即可。

例 32

```
syms x y z
f=x^2+x
[a0, an, bn]=mfourier(f)
```

结果为：

```
a0 =
2/3*pi^2

an =
((n^2*pi^2*sin(pi*n)-2*sin(pi*n)+2*pi*n*cos(pi*n)+n*cos(pi*n)
+pi*n^2*sin(pi*n))/n^3+(n^2*pi^2*sin(pi*n)-2*sin(pi*n)
+2*pi*n*cos(pi*n)-n*cos(pi*n)-pi*n^2*sin(pi*n))/n^3)/pi

bn =
(-(n^2*pi^2*cos(pi*n)-2*cos(pi*n)-2*pi*n*sin(pi*n)-n*sin(pi*n)
+pi*n^2*cos(pi*n))/n^3+(n^2*pi^2*cos(pi*n)-2*cos(pi*n)-2*pi*n*sin(pi*n)
+n*sin(pi*n)-pi*n^2*cos(pi*n))/n^3)/pi
```

由于 MatLab 不能像人一样将 $\cos n\pi$ 等化为$(-1)^n$的形式，因此所得结果显得很长。

6.6 多 重 积 分

多重积分其本质与普通积分一样，只是将沿一维的积分改为沿二维三维乃至多维的积分罢了。由于 MatLab 目前还没有一个多重积分的命令，因此本节还是沿用一维积分时所用的 int 命令，并结合对积分函数图像的观察，完成对多重积分的计算。

6.6.1 二重积分

在一个面上积分是二重积分的本质。只要能明确地将积分面表达出来并恰当转化成 int 命令中所需的积分限的形式，二重积分的结果就得到了。在前面，已经学习了 int 命令的各种使用方法，现在的重点是根据画出的积分平面的外形，正确地定出两组积分限。在此，将用 ezplot 命令画出积分平面外形。具体运算如下例。

例 33

计算函数 $f=x^2/y^2$ 在区域 D 上的积分，其中 D 为直线 $y=2x$，$y=x/2$，$y=12-x$ 围成的区域。

具体步骤如下：

(1) 划定积分区域

```
syms x y
f=x^2/y^2;
y1=2*x;
y2=x/2;
y3=12-x;
ezplot(y1)
hold on
ezplot(y2)
hold on
ezplot(y3, [-2 15])
```

3 条直线相应区域即为积分区域，如图 6.3 所示。

(2) 确定积分限

由图 6.3 可知，可将整个积分区域按 x 分为 xA-xB 和 xB-xC 两段。这样，在 xA-xB 段的积分下限为 y=x/2，积分上限为 y=2x; 在 xB-xC 段的积分下限为 y=x/2，积分上限为 y=12-x。下面分别求出 xA，xB 和 xC 的值，过程为：

```
pointA=fzero('2*x-x/2', 0)
pointB=fzero('2*x-(12-x)', 4)
pointC=fzero('12-x-x/2', 8)
```

求得结果为：

```
pointA = 0
pointB = 4
pointC = 8
```

即 xA=0，xB=4，xC=8。

到此，已经求得了积分所需的两个积分限。

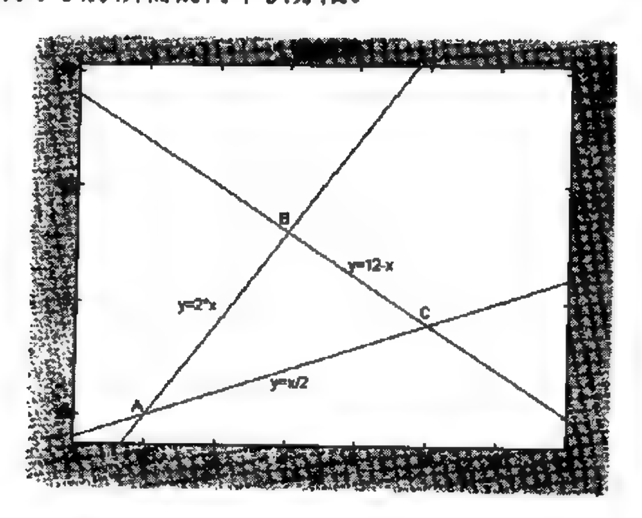


图 6.3 3 条直线相交区域即为积分区域

(3) 积分运算

按照前面思路，先在不同的 x 的区域内求出函数 f 在相应积分限内的对 y 的积分值 A1，A2，此值为 x 的函数；然后再让 A1，A2 对 x 进行积分，并将积分值相加得出最后的结果。具体操作为：

```
A1=int(f, y, x/2, 2*x)
A2=int(f, y, x/2, 12-x)
B1=int(A1, 0, 4)
B2=int(A2, 4, 8)
answer=B1+B2
```

结果为：

```
A1 =3/2*x
A2 =1/(-12+x)*x^2+2*x
B1 =12
B2 =120-144*log(2)
answer =132-144*log(2)。
```

本题最后结果即为：132−144log2。

6.6.2 三重积分

三重积分思路与二重积分相同，但在确定积分限时更加繁琐。在实际应用中，往往要根据积分区域的三维图形来确定积分限。

例 34　试求函数 $f(x, y, z)=x^2+y^2-z^2$ 在区域 D 上的积分，区域 D 为 $D=\{(x, y, z)|x^{2/3}+y^{1/2}+z^{2/5}\leqslant 1\}$

具体步骤如下：

(1) 划定积分区域

```
x=0: 3^.5/50: 3^.5;
y=0: 2/50: 2;
z=0;
for i=1: 51
   for j=1: 51
      z(j, i)=((1-(x(i)^2/3+y(j)^2/4))*5)^.5;
      if imag(z(j, i))<0
         z(j, i)=nan;
      end
      if imag(z(j, i))>0
         z(j, i)=nan;
      end
   end
end
mesh(x, y, z)
hold on
mesh(x, y, -z)
```

```
mesh(x, -y, z)
mesh(x, -y, -z)
mesh(-x, y, z)
mesh(-x, y, -z)
mesh(-x, -y, -z)
mesh(-x, -y, z)
```

函数的积分区域如图 6.4 所示。

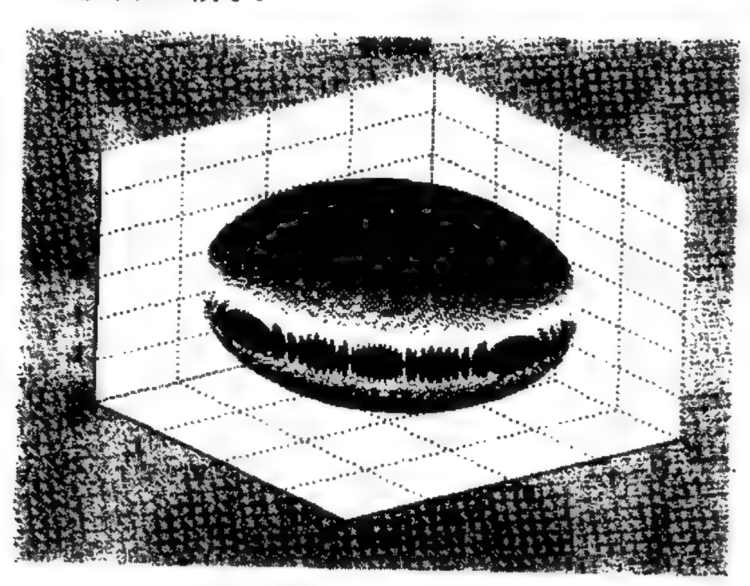


图 6.4　函数的积分区域

(2) 确定积分限

三重积分同普通积分一样，都要确定每一维的积分限。由图 6.4，只能知道整个三维积分区域的大体形状，要知道积分的上下限，还需换个角度看此图。

```
view(0, 90)          %沿 x 轴侧视
title('沿 x 轴侧视')
view(90, 0)          %沿 z 轴俯视
title('沿 z 轴俯视')
```

结果如图 6.5，6.6 所示。

从前几幅图知，由于区域 D 为一沿 3 轴均对称的空间几何体，因此在积分过程中可只计算第一象限，然后将结果乘以 8 即可。积分的顺序则可任选，此处先对 z 积分，然后再对 xy 平面积分。

```
syms x y z
f=x^2+y^2-z^2
Az=int(f, z, 0,  (5*(1-(x^2/3+y^2/4)))^0.5)
Azy=int(Az, y, 0, (4*(1-(x^2/3)))^0.5)
Azyx=int(Azy, 0, 3^0.5)
ANSWER=8*Azyx
```

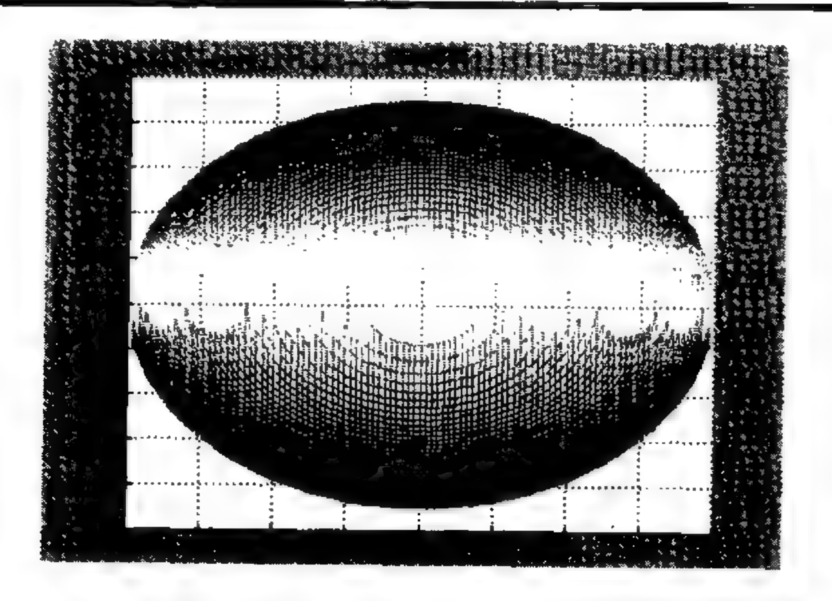


图 6.5 沿 X 轴侧视

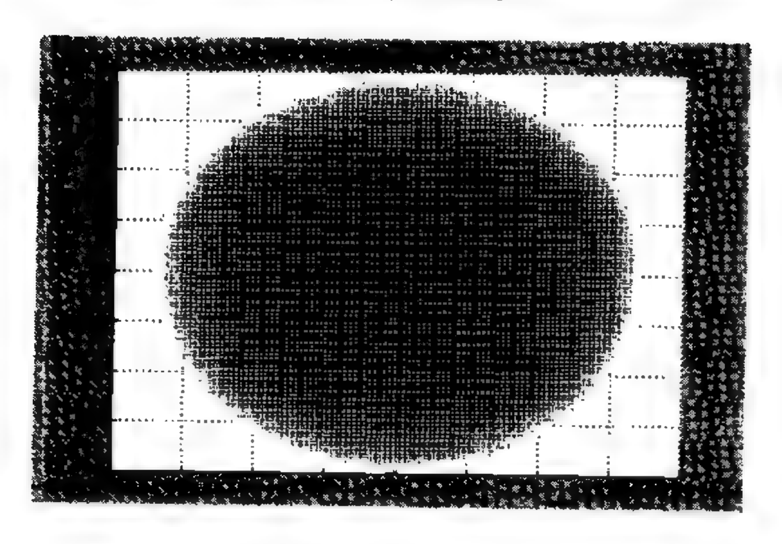


图 6.6 沿 Z 轴俯视

运算结果为：

```
Az =
1/6*x^2*(180-60*x^2-45*y^2)^(1/2)+1/6*(180-60*x^2-5*y^2)^(1/2)*y^2
-1/648*(180-60*x^2-45*y^2)^(3/2)

Azy =
-13/72*5^(1/2)*pi*x^4+7/12*5^(1/2)*pi*x^2-1/8*5^(1/2)*pi+13
/72*i*5^(1/2)*log(5)*x^4-7/12*i*5^(1/2)*log(5)*x^2+1/8*i*5^(1/2)*lo
g(5)+13/72*i*5^(1/2)*log(3)*x^4-7/12*i*5^(1/2)*log(3)*x^2+1/8*i*5^(1/2)
*log(3)-13/72*i*x^4*5^(1/2)*(log(3)+log(5))+7/12*i*5^(1/2)*x^2*(log(3)+
log(5))-1/8*i*5^(1/2)*(log(3)+log(5))
```

```
Azyx =
2/15*5^(1/2)*pi*3^(1/2)

ANSWER =
16/15*5^(1/2)*pi*3^(1/2)
```

本题最后结果为：ANSWER=16/15 • $5^{1/2}$ • $3^{1/2}\pi$。

6.7 符号方程及方程组的求解

一般在计算中遇到的代数方程组包括线性方程组、非线性方程组和超越方程组。在MatLab中，用于求解代数方程组的命令有两个：linsolve和solve命令。其中linsolve是专门用来求解线性方程组的命令，而solve命令的适用范围则是所有代数方程组。下面分别加以介绍。

6.7.1 求解线性方程组linsolve

linsolve命令对解形如AX=B的线性方程组运用自如。但在对矩阵A的运算中有如下限制：A必须至少是行满秩的。linsolve的具体使用格式为：

```
X=linsolve(A, B)
```

上面使用格式，得到方程组的特解X。

例 35

```
A=[34 8 4;3 34 6;3 6 8];
B=[4;6;2];
X=linsolve(A, B)
X =
[  66/955]
[ 144/955]
[ 106/955]
```

例 36

```
syms a b c d e f g h i m n p
A=[a b c;d e f;g h i];
B=[m;n;p];
X=linsolve(A, B)
X =
[-(b*f*p-i*b*n-c*p*e+c*h*n-m*h*f+i*m*e)/(h*a*f-h*d*c-g*b*f+i*d*b-i*a*e+g*c*e)]
[(a*f*p-i*a*n-c*d*p-f*g*m+i*d*m+n*g*c)/(h*a*f-h*d*c-g*b*f+i*d*b-i*a*e+g*c*e)]
[-1/(h*a*f-h*d*c-g*b*f+i*d*b-i*a*e+g*c*e)*(p*a*e-p*d*b+g*n*b-h*a*n-g*e*m+h*d*m)]
```

6.7.2 求解非线性方程组和超越方程组

solve 命令一般可解上面谈到的所有代数方程组，但有时会由于方程组结构上的原因而产生例外。另外，当方程组没有符号解且又无参数时，solve 命令最后会得出数值解。

下面是 solve 命令的几种具体使用格式。

1. solve(E)，solve(E, var)

在上述格式中，E 为符号方程，var 为待求符号变量。命令 solve(E)将给出方程 E 的关于其默认变量的符号解。当方程含有多个变量时，参数 var 将负责通知系统把哪个变量作为未知量来求解。

例 37

```
syms x a b c
f=sym('a*x^2+b*x+c=0');
x=solve(f)
```

结果为：

```
x =
[ 1/2/a*(-b+(b^2-4*a*c)^(1/2))]
[ 1/2/a*(-b-(b^2-4*a*c)^(1/2))]
```

例 38

```
syms x a b
g1=sym('sin(x)+tan(a)*b=0');
g2=sym('sin(x)*a+tan(a)*b=0');
g3=sym('sin(4)*a+tan(a+3)*5=0');
a1=solve(g1, a);
a2=solve(g2, a);
a3=solve(g3, a)
```

结果为：

```
a =
-atan(sin(x)/b)

a2=solve(g2, a);
??? Error using ==> solve
Error, (inallvalues/rootseq) cannot evaluate with symbolic coefficients
Error in ==> E: \toolbox\symbolic\@sym\solve.m
On line 49  ==> [varargout{1: max(1, nargout)}] = solve(S{: });

a3 =
-3.4854514938227066852523169031753
```

上例充分表明，当方程没有符号解且又无参数时，solve 命令最后会求出数值解。

2. [a_1, a_2, …, a_n]=solve(E_1, E_2, …, E_n) [a_1, a_1, …, a_n]=solve(E_1, E_2, …, E_n, var_1, var_2, …, var_n)

以上两个格式并无太大差别。式中 E_1, E_2, …, E_n 为代数方程组，参数 var_1, var_2, …, var_n 为对待求变量的说明，可有可无。因为 solve 命令要求方程的数目与变量数相同才能进行求解，而且解得的结果并不是按照在 solve 命令后的括号内 var_1，var_2，…，var_n 的顺序分别赋值给 a_1，a_1，…，a_n，而是按照 var_1，var_2，…，var_n 在字典中的顺序赋值给 a_1，a_1，…，a_n。即当 var_1 在所有的变量中以在字典中的顺序排在最后一个时，则在结果中，a_n 内才是变量 var_1 的解。

例 39

试解非线性方程组：

$$\begin{cases} a+b+x=y \\ 2ax-by=-1 \\ (a+b)^2=x+y \\ ay+bx=4 \end{cases}$$

```
E1=sym('a+b+x=y');
E2=sym('2*a*x-b*y=-1');
E3=sym('(a+b)^2=x+y');
E4=sym('a*y+b*x=4');
[a, b, x, y]=solve(E1, E2, E3, E4)
```

结果为：

```
a =
[  1]
[ -1]
b =
[  1]
[ -1]
x =
[  1]
[ -1]
y =
[  3]
[ -3]
```

例 40

试解超越方程组：

$x^x=2$

$xy+x=1$

```
E1=sym('x^x=2')
E2=sym('x*y+x=1')
```

```
[x, y]=solve(E1, E2)
```

结果为：

```
x =
log(2)/lambertw(log(2))
y =
-(log(2)-lambertw(log(2)))/log(2)
```

在解一般方程组中，还有以下两种简化命令格式可用。但在一般情况下，前面两例的标准格式更可靠些，使用时应慎重考虑。

例 41 试求如下方程组的解：

$$\begin{cases} u^2/a+v^2=0 \\ u+v=1 \\ a^2+5a=9 \end{cases}$$

```
[a, u, v]=solve('u^2/a+v^2=0', 'u+v=1', 'a^2+5*a=9')
```

结果为：

```
a =
[ -5/2+1/2*61^(1/2)]
[ -5/2+1/2*61^(1/2)]
[ -5/2-1/2*61^(1/2)]
[ -5/2-1/2*61^(1/2)]
u =
[ 23/26-1/26*61^(1/2)-1/13*(-2-5*61^(1/2))^(1/2)]
[ 23/26-1/26*61^(1/2)+1/13*(-2-5*61^(1/2))^(1/2)]
[ 23/26+1/26*61^(1/2)-1/13*(-2+5*61^(1/2))^(1/2)]
[ 23/26+1/26*61^(1/2)+1/13*(-2+5*61^(1/2))^(1/2)]
v =
[ 3/26+1/26*61^(1/2)+1/13*(-2-5*61^(1/2))^(1/2)]
[ 3/26+1/26*61^(1/2)-1/13*(-2-5*61^(1/2))^(1/2)]
[ 3/26-1/26*61^(1/2)+1/13*(-2+5*61^(1/2))^(1/2)]
[ 3/26-1/26*61^(1/2)-1/13*(-2+5*61^(1/2))^(1/2)]
```

当用符号表达式 S_1，S_2，…，S_n 代替 solve 命令中的符号方程组 E_1，E_2，…，E_n 时，就意味着要求以 $S_1=0$，$S_2=0$，…，$S_n=0$ 为方程组的解。

例 42 试求如下表达式组均等于零时的解。

$$\begin{cases} s_1=x^2+4xy+z; \\ s_2=x+3yz-3; \\ S_3=y+\sin z; \end{cases}$$

```
syms x y z a b c
S1=x^2+4*x*y+z;
S2=x+3*y*z-3;
S3=y+sin(z);
```

```
[x, y, z]=solve(S1, S2, S3)
```

结果为：

```
x =
-1.2488907023777540227165494966924
y =
-0.39875957366860047824596051700207
z =
-3.5517564826409285761070228228627
```

6.7.3 方程的数值求解方法

在符号运算中，由于运算比较复杂且符号变量占用空间较大，因此在求解大的符号方程及方程组时速度很慢。而数值求解因高效的算法和其所用变量占用空间小，所以在方程求解方面有很大的优势。当方程中无自由参变量，且不要求得到其符号解时，对于大型方程及方程组选用数值求解法则很快捷。

与前面符号方程及方程组的求解方法不同，数值求解法的两个命令 fzero 和 fsolve 的直接目的是求解函数的零点。因此，首先需对方程和方程组作一转化，如：方程 f(x)=g(x) 应转化为 F(x)=f(x)-g(x)，方程组中也如此。然后再将函数 F(x)写为 MatLab 的 m 函数，以便在 fzero 和 fsolve 命令中调用。

1．一元方程转化的函数，其零点的求法用 fzero 命令

命令 fzero 主要有如下 3 种使用格式：

```
z = fzero('fun', x)
z = fzero('fun', x, tol)
z = fzero('fun', x, tol, trace)
```

命令 z=fzero('fun', x)用来求解一个值域属于实数集的一元函数 fun 的零点。x 是对其零点的一个估计值，fzero 将在 x 附近进行寻找，其返回值 z 是靠近函数变号点的坐标值，也就是函数图像穿过 x 轴的那一点。注意：如果函数在 x_0 点只是与 x 轴相切，即函数值并未变号，则 fzero 命令将不能找出这一点，只能返回一个非数 NAN 或一个极大的数。

fzero('fun', x)中，估计值 x 可以是一个标量，也可以是一个二维向量。当 x 为标量时，fzero 命令将在 x 左右的一个有限邻域内查找零点，但如果在 z 很小的邻域内不只一个零点，fzero 也只能返回距离 x 最近的一个值。当 x 为一个形如$[x_1, x_2]$的向量时，则首先要求函数值 $f(x_1)f(x_2)<0$，然后 fzero 命令将严格在$[x_1, x_2]$的闭区间内寻找零点。如果未能找到，系统将给出提示。

例 43　试用 fzero 命令求解以下函数的零点

$f(x)=x^2$

$g(x)= x^2+4x-256$

具体步骤如下：

(1) 在 MatLab 内创建函数 f(x)，g(x)

```
function y=g(x)
```

```
y=x^2+4*x-256;

function y=f(x)
y=sin(x)+1;
```

为了方便估测零点位置，再准备一个函数 horizontal 用来表现 x=0 的水平线。

```
function y=horizontal(x)
y=0;
```

(2) 估计函数的零点位置

此处如函数简单如 f(x)，自然可看出大概位置，但如 g(x)，则一时看不出，此时可用 fplot 命令大概画出函数曲线，目测一下零点位置。

```
fplot('g', [-20, 20])
hold on
fplot('horizontal', [-20, 20])
```

画得结果如图 6.7 所示。

由图 6.7 可见，函数 g(x)的零点分别位于 x=-17，x=15 处。

(3) 求解函数零点

```
xf=fzero('f', -1.5);
xg1=fzero('g', -17);
xg2=fzero('g', [0, 15]);
```

结果为：

```
xf =
 -9.2234e+018
xg1 =
      -18.125
xg2 =
      14.125
```

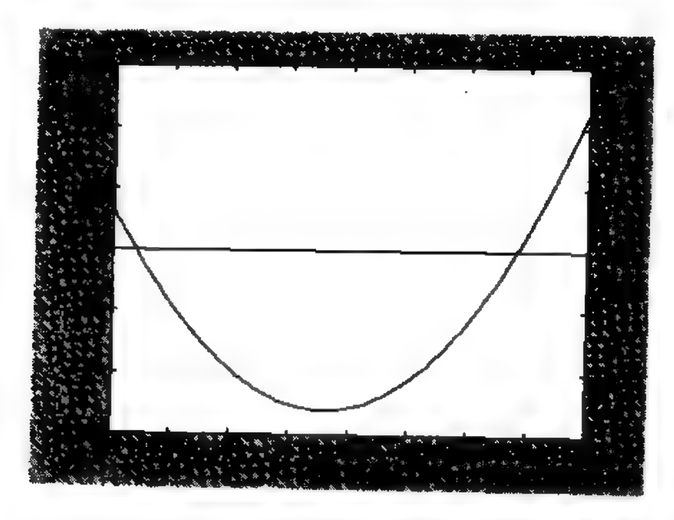


图 6.7 零点估计

在 fzero 命令的另外两个使用格式 z=fzero('fun'，x，tol)和 z= fzero('fun'，x，tol，trace)中，参数 tol 为 fzero 命令在进行迭代运算中的允许误差，其默认值为 eps=2.2204e-016。参数 trace 负责是否将计算中的迭代结果显示出来，当其为一非零值时，将显示结果。具体使用请见下例：

例 44 试求函数 $Fun=\sin x^2/x+xe^x-4$ 的零点

具体步骤如下：

(1) 创建函数 Fun

```
function y=Fun(x)
y=sin(x.^2)./x+x.*exp(x)-4;
```

注：式中运算符号前加“.”是为后面作图时用到的矩阵计算。

(2) 估测零点位置

```
x=0.01: 0.00001: 4;
y=Fun(x);
plot(x, y);
hold on
fplot('horizontal', [0, 4])
```

如图 6.8 所示，零点大概在 x=1.2 处。

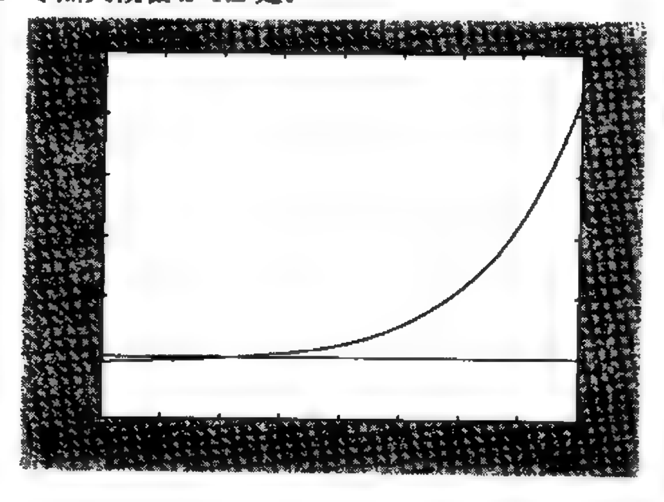


图 6.8 零点估计

(3) 求解并列出计算中间量

```
x=fzero('Fun', 1.2, [], 1)
 Func evals       x            f(x)          Procedure
    1            1.2         0.810356        initial
    2         1.16606         0.580807        search
    3         1.23394          1.04774        search
    4           1.152         0.48793        search
    5           1.248         1.14846        search
    6         1.13212          0.35872        search
```

```
       7        1.26788       1.29338       search
       8          1.104      0.180157       search
       9          1.296       1.50347       search
      10        1.06424    -0.0643217       search
      Looking for a zero in the interval [1.0642,  1.296]

      11        1.07374   -0.00670048       interpolation
      12        1.07484  3.98095e-006       interpolation
      13        1.07484  -2.0834e-009       interpolation
      14        1.07484 -4.44089e-016       interpolation
      15        1.07484  2.66454e-015       interpolation

   x =      1.0748
```

此时如将计算允许误差由 eps 改设为 0.1，结果变为：

```
   x=fzero('Fun', 1.2, 0.1, 1)
    Func evals      x            f(x)          Procedure
       1           1.2       0.810356         initial
       2        1.16606       0.580807         search
       3        1.23394        1.04774         search
       4          1.152        0.48793         search
       5          1.248        1.14846         search
       6        1.13212        0.35872         search
       7        1.26788        1.29338         search
       8          1.104       0.180157         search
       9          1.296        1.50347         search
      10        1.06424     -0.0643217         search

     Looking for a zero in the interval [1.0642,  1.296]
   x =      1.0642
```

2. 非线性方程组的求解 fsolve

命令 fsolve 的使用格式为：

```
X=fsolve('functions_name', X0)
```

式中 functions_name 是预先以 m 函数格式写入 MatLab 的函数组的函数名。X0 是当函数组均等于零时对各变量的解的估计。不过这时就没有什么绘图命令来帮助估测了。

例 45

试求如下方程组在 x，y，z 均等于 1 附近的解。

$$\begin{cases} \sin x+y+x^2e^z-4=0 \\ x+yx=0 \\ xyx=0 \end{cases}$$

具体步骤如下：

(1) 创建方程 Funs

```
function m=Funs(n)
x=n(1);
y=n(2);
z=n(3);
m(1)=sin(x)+y+z^2.*exp(x)-4;
m(2)=x+y*z;
m(3)=x*y*z;
```

(2) 解方程

```
n=fsolve('Funs', [1 1 1])
n =
   0.00014943 -7.4724e-005      1.9998
```

这个解很奇怪，x，y 比 z 要小很多，它是否是方程组的解呢?
可以验证一下:

```
m=Funs(n)
```

结果为:

```
m =
 -1.3152e-008 -3.1511e-009 -2.2331e-008
```

由于 fsolve 命令默认的所求解的精度为 0.0001，所解方程组成立精度也为 0.0001，所以 n 是方程组 Funs=0 在[1，1，1]附近的解。

6.8 习　　题

- 极限导数与微分

(1) 求函数 $y=(1+x)^{1/x}$ 在 $x=0$ 处的极限。
(2) 求函数 y=sin3x/tg5x 在 x=0 处的极限。
(3) 求函数 $y=nx/3^x$ 在趋向正无穷处的极限值。
(4) 求函数 $y=\ln^2x/x^3$ 在趋向正无穷处的极限值。
提示：用取绝对值函数 abs。
(5) 求函数 $y=1/x^2-3x+3$ 的 50 阶导数。
(6) 求函数 $y=a\sin(be^{ct}+t^a)$ 在 t=b 处的 3 阶导数。

- 积分

(1) 求下列不定积分:

```
1/sinx
1/sin^2x
1/sin^3x
```

```
1/(a^2-x^2)
(2-sinx)/sin^2x
((x^2-3)^1/2-(x^2+3)^1/2)/(x^4-9)^1/2
(ac+b)/(ac^2+bc+c)关于 c 的不定积分
```

(2) 求下列定积分及广义积分：

```
y=(x^2+a)^1/2              在[-2, 2]
y=(sinxcosx)^2             在[-π, π]
y=besselj(0, x)            在[1, 2]
```

- 化简、提取与替换带入

(1) 对下式分别用 simple 和 simplify 命令化简。

```
(cos3acosx);
x^2-3x+sin(acos3x);
sin2x/tgx;
```

(2) 对方程解进行提取与替换代入。

方程解为：

```
t=solve('a*x^6+b*x^2+c')
```

- 级数求和

(1) $(z-1)^n/(n^2 2^n)$　　$n=1\rightarrow\infty$

(2) $n(-1)^{n+1}z^n$　　$z\in C$，$n=1\rightarrow\infty$

(3) $(3n+1)(z-1)^n$　　$z\in C$，$n=1\rightarrow\infty$

- 泰勒、傅里叶级数展开

(1) 一元函数 taylor 展开

① $y=e^{-2x}$　　x=0 处 6 阶麦克劳林型泰勒展开式

② y= asin(bx)x/b　　b=a 处变量 b 的 4 阶麦克劳林型泰勒展开式

③ y= x/sinx　　x=2 处 20 阶麦克劳林型泰勒展开式

(2) 多元函数的完全泰勒展开

f=xyz　　(2，2，4)处 7 阶展开式

f=xy+ytgz　　(3，4，5) 处 4 阶展开式

f=sin(xyz)/x+ycos(zx)　　(10，20，30)处 10 阶展开式

(3) 傅里叶级数展开

① y=x

② $y=x^2$

③ $y=\sin(2x^2)$

- 符号方程及方程组的求解

(1) 求解线性方程组：AX=B

```
A=[45 23 53 65; 21 53 35 21;
7 23 14;  8  3  8  2];
B=[ 3 ; 4 ; 5 ; 6];
```

(2) 求解方程 f= sinx+tgx+1=0

(3) 求解方程组：

$$\begin{cases} x+y+z=0 \\ x^2+yz+x=10190 \\ x/y+z/y+y/x+y/z=16327/225 \end{cases}$$

第 7 章　多元函数分析及常微分方程

在第 6 章中，学习了高等数学的基本概念在 MatLab 中的用法，但前面介绍的这些方法只能解决一些基本的数学问题，如何运用这些命令来完成更深入的数学问题将是这一章的主要问题。在本章中，将讲述如何用 MatLab 解决诸如多元函数的求导、极值、空间曲线的有向积分、信号的基本处理和求解常微分方程等更加深入的方法。

在本章的学习中，不仅要学习具体数学问题的解法，更要注意解决问题的思路，以便以后用 MatLab 独立完成所遇到的新的问题。

7.1　多元函数的极限、微分、极值

7.1.1　多元函数的极限

一般来说，沿不同趋向路线计算某一点的极限，不是每个多元函数都有相同的值，因为有的函数在有些点并不存在极限。鉴于多元函数的极限与趋向路线有着非常密切的关系，而对趋向路线的数学描述又是灵活多样的，因此，MatLab 未能提供一个通用且功能强大的命令来求解多元函数的极限。但对于平时遇到的一般多元函数的极限问题，由于根据已经存在且与趋向路线没有任何关系，因此只需利用前面的求极限命令 limit 即可完成。

例 1　试求 $f=(x^2+y^2)/(\sin x+\cos y)$在点(0，$\pi$)处，

$g=(x+yz+e^{\sin(xz)})/(x+y/z)$在点(1，2，3)处的极限值

```
syms x y z
f=(x^2+y^2)/(sin(x)+cos(y));
g=(x+y*z+exp(sin(x*z)))/(x+y/z);
LIMIT1=limit(f, 0)
LIMITF=limit(LIMIT1, pi)
LIMITG=limit(limit(limit(g, 1), 2), 3)
```

结果为：

```
LIMIT1 =
y^2/cos(y)
LIMITF =
-pi^2
LIMITG =
21/5+3/5*exp(sin(3))
```

7.1.2　多元函数的求导

附：梯度的数值求法

在多元函数微分学的导数部分中，分别有方向导数、偏导数和梯度 3 个既有联系又有区别的概念。在这 3 个导数概念中，方向导数和偏导数是梯度的基础，而梯度又是前面 2 概念的总结和提高。当一个多元函数在某一点的梯度已经知道，那这个函数在此点的偏导数和方向导数也就知道了，因此，在 MatLab 中，求解梯度是计算的关键和重点，有了对梯度的运算，其他两个的求解则迎刃而解。关于这 3 个概念，请参阅有关的书籍。

1. 梯度(jacobian)

在 MatLab 中，求解一个多元函数梯度的命令为 jacobian，它的使用格式为：

```
jacobian(f)
```

其中 f 为多元函数的表达式。
jacobian 命令是一个功能非常强大的求导命令。在此只用到了它很小的一部分功能。
例 2

```
syms x y z
f=log(z^(-y/x-x/y));
gradf=jacobian(f)
g=2*x*y/(x^2+y^2+z^2);
gradg=jacobian(g)
h=x^2*y+y^2*z+z^2*x;
gradf=jacobian(f)
```

结果为：

```
gradf =
[(y/x^2-1/y)*log(z),  (-1/x+x/y^2)*log(z),  (-y/x-x/y)/z]
gradg =
[2*y/(x^2+y^2+z^2)-4*x^2*y/(x^2+y^2+z^2)^2, 2*x/(x^2+y^2+z^2)
-4*x*y^2/(x^2+y^2+z^2)^2, -4*x*y/(x^2+y^2+z^2)^2*z]
gradh =
[ 2*x*y+z^2,  x^2+2*y*z,  y^2+2*x*z]
```

2. 偏导数

用 jacobian(f)命令求出多元函数 f 的梯度，它的第 1 分量、第 2 分量、第 3 分量分别是函数 f 的(x 方向、y 方向、z 方向)偏导数。

3. 方向导数

多元函数在某一点沿单位向量 v 的增长率即为函数在在此点沿方向 v 的导数，称为方向导数。在过此点的所有方向导数中，模数最大的所对应的方向的单位向量即为多元函数

在此点的梯度。

根据定义，多元函数的梯度点乘单位向量 v 的结果即为多元函数沿此方向上的方向导数。这样，在 MatLab 中，方向导数的计算方法为：

```
jacobian(f)*v
```

例 3

试求 7.2 例中多元函数

f 沿 v1=$((1/3)^{1/2}$，$(1/3)^{1/2}$，$(1/3)^{1/2})$的方向导数；

g 沿 v2=$(0$，$(2/5)^{1/2}$，$(2/5)^{1/2})$的方向导数；

h 沿 v3=(3/5，4/5，0)的方向导数。

具体操作为：

```
v1=[1/3^0.5,  1/3^0.5,  1/3^0.5]';
v2=[0, 2/5^0.5, 1/5^0.5]';
v3=[3/5, 4/5, 0]';
ansf=gradf*v1
ansg=gradg*v2
ansh=gradh*v3
```

结果为：

```
ansf =
1/3*(y/x^2-1/y)*log(z)*3^(1/2)+1/3*(-1/x+x/y^2)*log(z)*3^(1/2)+1/3*(
-y/x-x/y)/z*3^(1/2)
ansg =
2/5*(2*x/(x^2+y^2+z^2)-4*x*y^2/(x^2+y^2+z^2)^2)*5^(1/2)-4/5*x*y/(x^2
+y^2+z^2)^2*z*5^(1/2)
ansh =
6/5*x*y+3/5*z^2+4/5*x^2+8/5*y*z
```

附：　求梯度的数值命令

在 MatLab 求梯度的命令中，除了用来求解符号函数梯度的 jacobian 命令外，还有两个在实数矩阵中求解梯度的命令：grdient 和 surfnorm。

```
gradient
```

该命令用于求解由矩阵 F 表示场的梯度向量。其使用格式为：

1. FX = gradient(F)
2. [FX，FY] = gradient(F)
3. [Fx，Fy，Fz，...] = gradient(F)
4. [...] = gradient(F，h)
5. [...] = gradient(F，h1，h2，...)

第 1 种格式用来求解一维向量 F 的数值梯度 dy/dx，在此格式中取步长 dx=1。

第 2 种和第 3 种格式的作用与第 1 种格式相同，只不过它们是求解二维矩阵及多维矩

阵F的数值梯度．并且其各个步长dx，dy，dz，...均取为1。

例 4

```
x=0: 6;
y=1/2*x.^2;
plot(y, 'r')
hold on
dyx=gradient(y);
plot(dyx, 'b')
```

结果如图7.1所示。

```
plot(y) and plot(dyx, 'b')
```

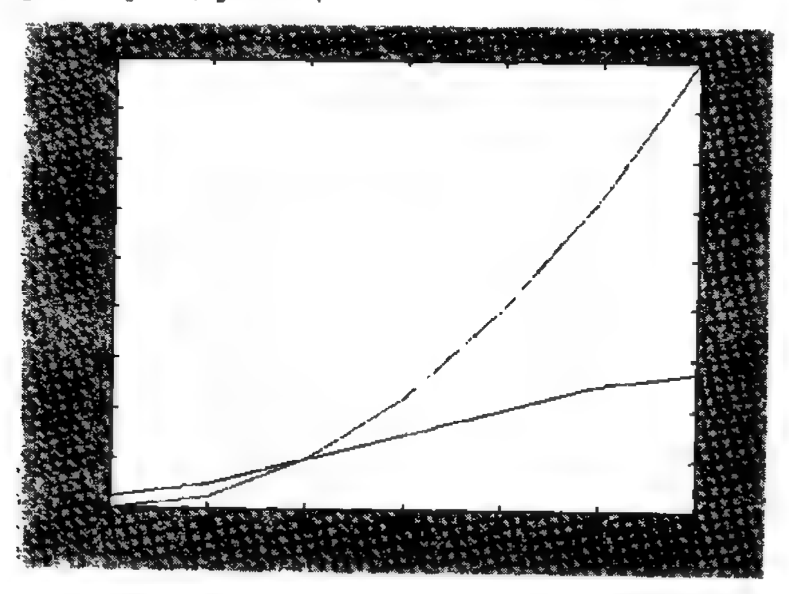


图 7.1　函数 $y=1/2x^2$ 及其导数 $y=x$ 的图像

由于算法原因，向量y=1/2x.^2的导数y'=x在两端与实际值有所出入。

第4、5种使用格式在计算矩阵F的梯度时较前3种格式加设了每个方向(x，y，z…)上求导的步长h1，h2，h3，…，这种使用格式能更精确地表现出矩阵F在每个位置的梯度值，因此使用得最多。

例 5

```
x=0: pi/10: 4*pi;
y=sin(x);
dyx=gradient(y, pi/10);
plot(x, y, 'r')
hold on
plot(x, dyx, 'b');
```

可查看实际计算结果dyx，也可通过图7.2观察计算结果是否准确。

```
plot(y) and plot(dyx, 'b')
```

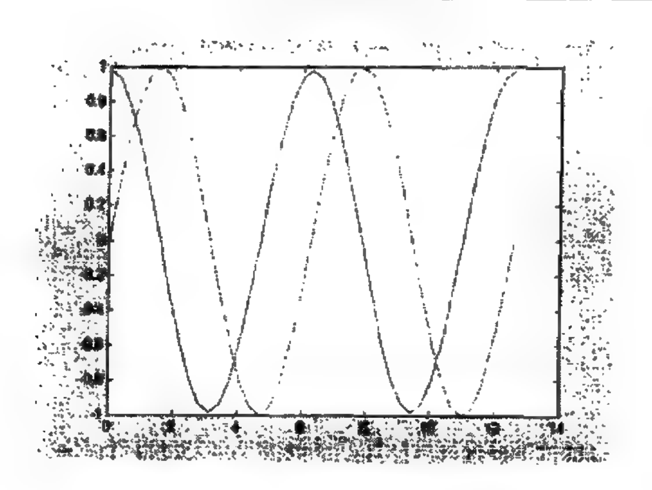


图 7.2 函数 y=sinx 及其导数 y=cosx 的图像

例 6 求 peaks 函数 z=3(1-x).^2.*exp(-(x.^2)-(y+1).^2)-10(x/5-x.^3-y.^5).*exp (-x.^2 -y.^2) -1/3exp(-(x+1).^2-y.^2)在平面{x，y|-3≤x≤3，-3≤y≤3}s 上的数值梯度

```
[x, y]=meshgrid(-3: 0.3: 3);
z=3*(1-x).^2.*exp(-(x.^2)-(y+1).^2)-10*(x/5-x.^3-y.^5).*exp(-x.^2-y.
^2)-1/3*exp(-(x+1).^2-y.^2);
[dx, dy]=gradient(z, 0.1, 0.1);
```

由于求解得的 dx，dy 全是抽象的数字，很难从数值上看懂它们，下面用一个作图命令 quiver 将其表现在图中，至于 quiver 命令，将在 9.4 节详细介绍。

```
quiver(dx, dy)
```

得结果如图 7.3 所示。

```
quiver(dx, dy)
```

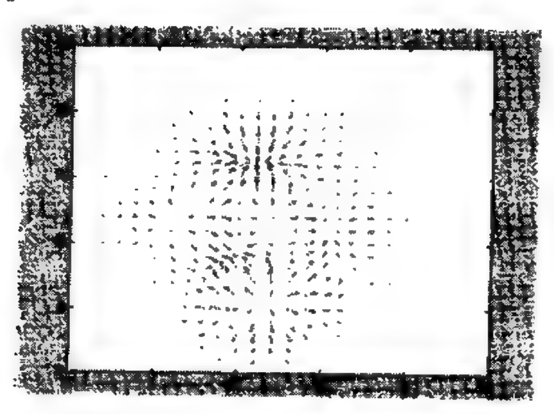


图 7.3 函数 peaks 的平面矢量图

2. surfnorm

该命令严格地说是用于求解在 mesh(Z)图像中每个节点处的法向量。其使用格式为：

```
[Nx, Ny, Nz] = surfnorm(Z)
[Nx, Ny, Nz] = surfnorm(X, Y, Z)
surfnorm(Z)
surfnorm(X, Y, Z)
```

上面前两种格式将求出每个节点处法向量的数值表达[Nx，Ny，Nz]。当格式为 surfnorm(Z)时，系统将把计算时所需的步长设为 1；格式为 surfnorm(X，Y，Z)时，系统将按 X，Y 的值计算步长。

后面两个命令格式将不返回数值值，而是自动先用 surf(Z)和 surf(X，Y，Z)命令画出曲面，然后在每个节点处以箭头的形式标出该点的法向量。

例 7

```
[x, y]=meshgrid(-2: 0.25: 4);
z=3*(1-x).^2.*exp(-(x.^2)-(y+1).^2)-10*(x/5-x.^3-y.^5).*exp(-x.^2-y.^2)-1/3*exp(-(x+1).^2-y.^2);
[Nx, Ny, Nz]=surfnorm(x, y, z);
surfnorm(x, y, z)
```

用户可以自己检查影响[Nx，Ny，Nz]的数值结果，由 surfnorm(x，y，z)绘得的向量图如图 7.4 所示。

```
surfnorm(x, y, z)
```

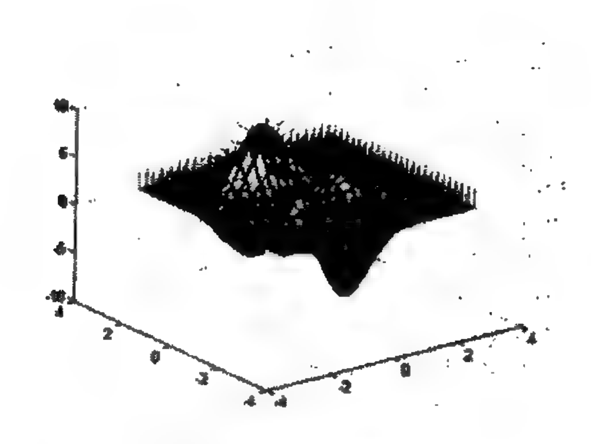


图 7.4 函数 peaks 的表面法向量图

7.1.3 隐函数的求导

无论是一元隐函数还是多元隐函数的求导，如果不是按照显函数的求导方法，MatLab 将不能识别。按照高等数学中的求解方法，这部分的数学表达如下：

设 n+1 元函数 $F(x_1, \cdots, x_{n+1})$在点$(x_1^0, \cdots, x_{n+1}^0) \in R^{n+1}$某个邻域 W 中有定义并且满

足下列条件：

$F(x_1^0, ..., x_{n+1}^0)=0$；

$F\in C^q(W)(q>1)$；

$F(x_1, ..., x_{n+1})$在点$(x_1^0, ..., x_{n+1}^0)$对x_{n+1}的导数不等于0。

则存在点$(x_1^0, ..., x_n^0)\in R^n$的一个邻域U，以及定义在U上的n元函数$x_{n+1}=f(x_1, ..., x_n)$使得下列结论成立：

$x_{n+1}^0=f((x_1^0, ..., x_n^0))$，$F(x_1, ..., x_n, f(x_1, ..., x_n))\equiv 0$；$((x1, ..., xn)\in U)$。

$f\in C^q(U)$。

$f(x_1, \cdots, x_n)$对x_i的导数等于负的$F(x_1, \cdots, x_n, f(x_1, ..., x_n))$对$x_i$的偏导数除以$F(x_1, ..., x_n, f(x_1, \cdots, x_n))$对$x_{n+1}$的偏导数。$(i=1, \cdots, n)$

上述定理提供了一个非常方便的求导方法，即在保证定理中的3个前提下，隐函数的求导可表达为：

$$\frac{\partial Xn+1}{\partial Xi}=-\frac{\frac{\partial F}{\partial Xi}}{\frac{\partial F}{\partial Xn+1}}$$

有了这个公式，多元函数的计算就可简单地完成了。

在具体计算中，可以用 jacobian 命令也可用 diff 命令，都可得到正确结果。现以 diff 命令为例解一题。

例 8

已知$x+y+z=e^{-(x+y+z)}$，试求z关于x的导数和z关于y，x的二重导数。

```
syms x y z
F=x+y+z-exp(-(x+y+z));
diffFx=diff(F, x)
diffFz=diff(F, z)
diffzx=-diffFx/diffFz
diffZXy=diff(diffzx, y)
diffZXz=diff(diffzx, z)
diffZxy=-diffZXy/diffZXz
```

结果为：

```
diffFx =
1+exp(-x-y-z)
diffFz =
1+exp(-x-y-z)
diffzx =
(-1-exp(-x-y-z))/(1+exp(-x-y-z))
diffZXy =
exp(-x-y-z)/(1+exp(-x-y-z))+(-1-exp(-x-y-z))/(1+exp(-x-y-z))^2*exp(-
x-y-z)
```

```
    diffZXz =
    exp(-x-y-z)/(1+exp(-x-y-z))+(-1-exp(-x-y-z))/(1+exp(-x-y-z))^2*exp(-
x-y-z)
    diffZxy =
    (-exp(-x-y-z)/(1+exp(-x-y-z))-(-1-exp(-x-y-z))/(1+exp(-x-y-z))^2*exp
(-x-y-z))/(exp(-x-y-z)/(1+exp(-x-y-z))+(-1-exp(-x-y-z))/(1+exp(-x-y-z))^
2*exp(-x-y-z))
```

以上是单个隐函数的求导计算，多元隐函数组的求导计算则更为复杂。现只提供三元隐函数组的求导定理，其具体运算则与前面的单个隐函数的求导运算基本相同。

三元函数组的求导定理：

设三元函数 F(x，y，z)，G(x，y，z)在点(x0，y0，z0)的某个邻域 W 中有定义，并且满足下列条件：(以下 detM 表示矩阵 M 的行列式)

F(x0，y0，z0)=0；G(x0，y0，z0)=0，

F，G∈Cq (W)

$$\det\frac{\partial(F,G)}{\partial(y,x)}\bigg|_{(x_0,y_0,z_0)}\neq 0$$

则存在以 x0 为中心的一个区域(a，b)以及定义在(a，b)上的两个函数 y=f(x)，z=g(x)满足如下条件：

y0=f(x0)；z0=g(x0)；F(x，f(x)，g(x))≡0；(a<x<b)

函数 f，g 在(a，b)上 q 阶连续可微；

$$\begin{bmatrix}\frac{dy}{dx}\\ \frac{dz}{dx}\end{bmatrix}=-\left[\frac{\partial(F,G)}{\partial(y,x)}\right]^{-1}\begin{bmatrix}\frac{\partial F}{\partial x}\\ \frac{\partial G}{\partial x}\end{bmatrix}$$

根据上面的定理，试完成下面的练习。

(1) F(x，y，z)=$x^2+y^2+z^2-1=0$，G(x，y，z)=$x^2+y^2-2z^2-1=0$，试求 y，z 关于 x 的导数。

(2) x=ucosv，y=usinv，z=uv，试求 y，z 关于 x 的导数。

7.1.4 多元函数的局部极值

为了保证对命令和程序说明的针对性，在介绍多元函数的局部极值的求解前，先介绍一下以下几个数学定理和公式。

- 多元函数的局部极值定义：

设函数 f 在点 $x_0\in R^n$ 的某个邻域 U 中有定义，并且对于所有的 x∈U 都有

$f(x)\leqslant f(x_0)$ ($f(x)\geqslant f(x_0)$)

则称 f 在点 $x_0\in R^n$ 达到局部极大(极小)值，并称 x_0 为 f 的一个局部极值点。

- 多元函数的二阶 Taylor 公式：

$f(x_1,x_2,\ldots\ldots,x_n)-f(x_1^0,x_2^0,\ldots\ldots,x_n^0)=$

$$\frac{1}{2!}(x_1-x_1^0,x_2-x_2^0,\ldots\ldots,x_n-x_n^0)\begin{bmatrix}\frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1\partial x_2} & \cdots\cdots & \frac{\partial^2 f}{\partial x_1\partial x_n}\\ \frac{\partial^2 f}{\partial x_2\partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots\cdots & \frac{\partial^2 f}{\partial x_2\partial x_n}\\ \cdots\cdots & \cdots\cdots & \cdots\cdots & \cdots\cdots\\ \frac{\partial^2 f}{\partial x_n\partial x_1} & \frac{\partial^2 f}{\partial x_n\partial x_2} & \cdots\cdots & \frac{\partial^2 f}{\partial x_n^2}\end{bmatrix}\begin{bmatrix}x_1-x_1^0\\ x_2-x_2^0\\ \cdots\cdots\\ x_n-x_n^0\end{bmatrix}+O(\sum_{i=1}^{n}(x_i-x_i^0)^2)$$

x_0 是 f 的局部极值点的必要条件是 $\mathrm{grad}f(x_0)=0$。

在多元函数的二阶 Taylor 公式中的 n×n 阶的 f 的导数矩阵称为 Hessian 矩阵，应用矩阵正定和负定的充分条件可以得到以下结论：

当 $A>0$，$AC-B^2>0$ 时，f 在点 $M_0(x_0, y_0)$取局部极小值；

当 $A<0$，$AC-B^2>0$ 时，f 在点 $M_0(x_0, y_0)$取局部极大值；

当 $AC-B^2<0$ 时，f 在点 $M_0(x_0, y_0)$不取局部极值，这时称 $M_0(x_0, y_0)$为 f 的一个鞍点；

当 $AC-B^2=0$ 时，需要进一步讨论。

由上可见，一般情况下求解多元函数 f 的步骤为：

(1) 由 gradf=0 求解函数 f 的驻点。

(2) 计算每个驻点的 A，B，C 值，以确定该点是否为极值点，是什么极值及其具体数值。

以上就是求解多元函数的局部极值的原理及步骤，具体应用，请见下例。

```
syms x y z lambda1 lambda2
f=2*x^4+y^4-2*x^2-2*y^2;
v=[x y];
F=jacobian(f, v).';
[X Y]=solve(F(1), F(2))

Hessian=maple('hessian(2*x^4+y^4-2*x^2-2*y^2, x, y)');
Leng=length(Hessian);
Hessian=sym(Hessian(8: Length-1));
```

在求得函数的 Hessian 矩阵后，下一步就是判断在各个点上是正定还是负定。

```
m=length(X);
q=0
for i=1: m
    x=X(i); y=Y(i);
    he=subs(Hessian);
    A=eig(he);
    nflag=1;
    gflag=1;
    for j=1: length(A)
        if double(A(j))<0
            gflag=0;
```

```
                break;
            end
        end
        for j=1: length(A)
                    if double(A(j))>0
                nflag=0;
                break;
            end
        end
        for j=1: length(A)
            if A(j)= =0
                gflag=0;
                nflag=0;
            end
        end
        if gflag==1
            q=q+1
            x
            y
            'is a max point'
        elseif nflag==1
            q=q+1
            x
            y
            'is a min point'
        end
    end
```

7.1.5 条件极值

在实际应用中更常见的是约束极值，即条件极值问题。例如，考虑以下两个问题：

Q1　　minf(x，y，z)　& maxf(x，y，z)
　　　G=g(x，y，z)=0
Q2　　minf(x，y，z)　& maxf(x，y，z)
　　　G1=g1(x，y，z)=0
　　　G2=g2(x，y，z)=0

问题Q1的几何意义是：在曲面S：{(x，y，z)|g(x，y，z)=0}上求函数f(x，y，z)的最小(大)值。问题Q2的几何意义是：在由g1(x，y，z)=0，g2(x，y，z)=0确定的曲线L上求函数f(x，y，z)的最小(大)值。

对上述问题，理论上可以由约束条件解出一个变量，如z=z(x, y)，然后带入到函数f(x，y，z)中，这样问题就由条件极值问题转化为两个变量的无条件的局部极值问题了，就可以用上面的方法求解了。但这样做会遇到非常繁琐的计算，因此，下面采用Lagrange乘子方法，先引入若干乘子，然后再将条件极值问题转化为无条件极值问题来进行求解。下面依次解决前面提出的两个问题。

对于问题 Q1，有如下定理可作 Lagrange 乘子方法的依据：

设 M0(x_0，y_0，z_0)是问题 Q1 的一个解(即满足条件 G=g(x，y，z)=0 下的函数 f(x，y，z)在 M0(x_0，y_0，z_0)达到极值)，又设 gradg(M0)≠0，则存在常数 lambda 使得：

```
gradf(M0)=lambda*gradg(M0)
```

上式中包括 3 个方程：

$$\frac{\partial f(x_0,y_0,z_0)}{\partial x}-\lambda\frac{\partial g(x_0,y_0,z_0)}{\partial x}=0$$
$$\frac{\partial f(x_0,y_0,z_0)}{\partial y}-\lambda\frac{\partial g(x_0,y_0,z_0)}{\partial y}=0$$
$$\frac{\partial f(x_0,y_0,z_0)}{\partial z}-\lambda\frac{\partial g(x_0,y_0,z_0)}{\partial z}=0$$

再与约束方程

```
g(x0, y0, z0)=0
```

联合就可解出 lambda，x_0，y_0，z_0。

例 9　试求 f(x，y)= y x^2+y^5/x 在曲线 g(x，y)=xy+x^2/y 上的极值

```
syms x y lambda
f=x^2*y+y^5/x;
g=x*y+x^2/y;
v=[x y];
F=jacobian(f, v);
G=jacobian(g, v);
A=F.'-lambda*G.';
A(3)=g;
[lambda x y]=solve(A(1), A(2), A(3))
f
fmin_max=subs(f)
```

结果为：

```
lambda =
[ 11/5]
[ 11/5]
x =
[ -3/5]
[ -3/5]
y =
[  1/5*15^(1/2)]
[ -1/5*15^(1/2)]
f =
x^2*y+y^5/x
fmin_max =
[ -6/125*15^(1/2)]
[  6/125*15^(1/2)]
```

例 10　用铁板制作一个容积为 V 的无盖长方体的盒子，试问怎样才能用料最省？

设长方体的长、宽、高分别为 x，y，z，则由题可知所用铁板面积为：f(x，y，z)=xy+2xz+2yz。令 g(x，y，z)=xyz−V。则问题化为解条件极值问题。

```
minf(x, y, z)=min(x*y+2*x*z+2*y*z)
g(x, y, z)=x*y*z-V=0
syms x y z V lambda
f=x*y+2*x*z+2*y*z;
g=x*y*z-V;
v=[x, y, x];
F=jacobian(f, v);
G=jacobian(g, v);
A=F.'-lambda*G.';
A(4)=g;
[lambda x y z]=solve(A(1), A(2), A(3), A(4))
f
fmin=subs(f)
```

结果为：

```
lambda =
(y^3+2*V)/y/V
x =
y
y =
y
z =
V/y^2
f =
x*y+2*x*z+2*y*z
fmin =
y^2+4/y*V
```

此时结果并没有给出具体的 x，y，z，lambda 和 f 的数值，这与 MatLab 中的算法有关，因此还要自己进行运算。

```
diffmin=diff(fmin)
S=solve(diffmin)
```

以上的运算结果为：

```
diffmin =
2*y-4*V/y^2
S =
[                                  2^(1/3)*V^(1/3)]
[ -1/2*2^(1/3)*V^(1/3)+1/2*i*3^(1/2)*2^(1/3)*V^(1/3)]
[ -1/2*2^(1/3)*V^(1/3)-1/2*i*3^(1/2)*2^(1/3)*V^(1/3)]
```

S 为函数 f 的最小值 fmin 的导数 diffmin=0 时的 3 个根，其中有 1 个实数，2 个复数。

由题意知，y 只能取实数，所以有

```
y=S(1);
fmin=subs(fmin)
```

结果为：

```
fmin=3*2^(2/3)*V^(2/3)
```

关于问题 Q2，有如下定理可作为 Lagrange 乘子方法的依据：

设 $M0(x_0, y_0, z_0)$是问题 Q2 的一个解(即满足条件 G1=g1(x，y，z)=0，G2=g2(x，y，z)=0 下的使函数 f(x，y，z)在 $M0(x_0, y_0, z_0)$达到极值)，又设在 $M0(x_0, y_0, z_0)$点的两个梯度向量 gradg1(M0)， gradg2(M0)不共线，则存在两个常数 lambda1，lambda2 使得：

```
gradf(M0)=lambda1*gradg1(M0)+ lambda2*gradg2(M0)
```

同样，上式包含了 3 个方程：

$$\frac{\partial f(x_0,y_0,z_0)}{\partial x}-\lambda 1\frac{\partial g1(x_0,y_0,z_0)}{\partial x}-\lambda 2\frac{\partial g2(x_0,y_0,z_0)}{\partial x}=0$$

$$\frac{\partial f(x_0,y_0,z_0)}{\partial y}-\lambda 1\frac{\partial g1(x_0,y_0,z_0)}{\partial y}-\lambda 2\frac{\partial g2(x_0,y_0,z_0)}{\partial y}=0$$

$$\frac{\partial f(x_0,y_0,z_0)}{\partial z}-\lambda 1\frac{\partial g1(x_0,y_0,z_0)}{\partial z}-\lambda 2\frac{\partial g2(x_0,y_0,z_0)}{\partial y}=0$$

这 3 个方程再与两个约束方程联立，

$g1(x_0, y_0, z_0)=0$

$g2(x_0, y_0, z_0)=0$

就可解得 x_0，y_0，z_0，lambda1，lambda2。

例 11

试求 f(x，y，z)=xyz 在两个平面 x+y+z−30=0 与 x+y−z=0 的交线上的最大值。

```
syms x y z lambda1 lambda2
f=x*y*z;
g1=x+y+z-30;
g2=x+y-z;
v=[x y z];
F=jacobian(f, v);
G1=jacobian(g1, v);
G2=jacobian(g2, v);
A=F.'-lambda1*G1.'-lambda2*G2.';
A(4)=g1;
A(5)=g2;
[lambda1 lambda2 x y z]=solve(A(1), A(2), A(3), A(4), A(5))
f
fmax=subs(f)
```

结果为：

```
lambda1 =
675/8
lambda2 =
225/8
x =
15/2
y =
15/2
z =
15
f =
x*y*z
fmax =
3375/4
```

7.1.6　显式复合函数微分求导

1．一元复合函数

在高等数学中，一元复合函数微分法的链式法则为：

设 y=f(u)，u=g(x)，u0=g(x0)，如果 f 在点 u0 可微，g 在 x0 可微，则复合函数 y=f(g(x)) 在点 x0 可微，并且：

$$\left.\frac{dy}{dx}\right|_{x_0}=\left.\frac{dy}{du}\right|_{u_0}\cdot\left.\frac{du}{dx}\right|_{x_0}$$

因此，根据以上链式法则，应用 diff 命令可以求得一元复合函数的导数。

例 12

已知：$y=\sin(u^3)$；$u=x^2tgx$。试求： x 对 y 的导数。

运算命令如下：

```
syms x u
u=x^2*tan(x);
y=sin(u^3);
diffyx=diff(y, x)
```

结果为：

```
diffyx =
cos(x^6*tan(x)^3)*(6*x^5*tan(x)^3+3*x^6*tan(x)^2*(1+tan(x)^2))
```

而将命令变为：

```
syms x u
y=sin(u^3);
u=x^2*tan(x);
diffyx=diff(y, x)
```

则结果为：

```
diffyx =0
```

由此例可见，在 MatLab 中，只要按已知顺序给出函数表达式，则 diff 命令将自动按照链式法则进行求导运算。

2. 多元复合函数

多元复合函数的求导在方法上同一元复合函数一样。当只需求解函数的几个有限的变量的导数时，用 diff 命令同样奏效。

例 13

试求多元复合函数：

$y_1=u_1u_2-u_1u_3$；$y_2=u_1u_3-u_2^2$；$u_1=x\cos y+(x+y)^2$；$u_2=x\sin y+xy$；$u_3=x^2-xy+y^2$；

y_1 关于 x，y_2 关于 y 的在点(0，1)处的导数。

具体命令为：

```
syms u1 u2 u3 x y
u1=x*cos(y)+(x+y)^2;
u2=x*sin(y)+x*y;
u3=x^2-x*y+y^2;
y1=u1*u2-u1*u3;
y2=u1*u3-u2^2;
diffy1x=diff(y1, x)
diffy2y=diff(y2, y)
x=0;
y=1;
diffy1x=subs(diffy1x)
diffy2y=subs(diffy2y)
```

结果为：

```
diffy1x =
(cos(y)+2*x+2*y)*(x*sin(y)+x*y)+(x*cos(y)+(x+y)^2)*(sin(y)+y)-(cos(y
)+2*x+2*y)*(x^2-x*y+y^2)-(x*cos(y)+(x+y)^2)*(2*x-y)
diffy2y =
(-x*sin(y)+2*x+2*y)*(x^2-x*y+y^2)+(x*cos(y)+(x+y)^2)*(-x+2*y)-2*(x*s
in(y)+x*y)*(x*cos(y)+x)
diffy1x =
     0.30117
diffy2y =
     4
```

以上的算法只适用于求解 1、2 个复合多元函数的关于 1、2 个变量的导数，若是用来求解一复合多元函数组的关于所有变量的导数，则相当麻烦。下面介绍另一种方法。

在介绍这种方法前，先讲求解一个多元函数组的关于所有变量的导数的命令 jacobian。它正是前面用来求解多元函数梯度的命令。

在具体介绍 jacobian 命令前，先介绍一下高等数学中的一个相关内容：

考察 m 个变元 u_1，u_2，…，u_m 的 k 个函数：

```
y1=f1(u1, u2, ..., um);
y2=f2(u1, u2, ..., um);
................
yk=fk(u1, u2, ..., um).
```

知道这 k 个函数均在某个区域 D(D 属于 Rm)有定义，则上面 k 个方程可以看成是由 D 到 Rk 的一个映射，记作：

```
y=y(u)
```

其中 $y=(y_1, y_2, \ldots, y_k)T$ 属于 Rk；$u=(u_1, u_2, \ldots, u_m)$ T 属于 Rm；$f=(f_1, f_2, \ldots, f_k)T$。如果所有函数 f_1，f_2，….，f_k 均在点 u_0 可微，则称由偏导数构成的矩阵

$$\left.\frac{\partial(y_1,y_2,\ldots,y_k)}{\partial(u_1,u_2,\ldots,u_k)}\right|_{u_0}=\begin{bmatrix}\frac{\partial y_1}{\partial u_1} & \frac{\partial y_1}{\partial u_2} & \cdots & \frac{\partial y_1}{\partial u_m}\\ \frac{\partial y_2}{\partial u_1} & \frac{\partial y_2}{\partial u_2} & \cdots & \frac{\partial y_2}{\partial u_m}\\ \cdots & \cdots & \cdots & \cdots\\ \frac{\partial y_k}{\partial u_1} & \frac{\partial y_k}{\partial u_2} & \cdots & \frac{\partial y_k}{\partial u_m}\end{bmatrix}_{u_0}$$

为 fi 函数组在点 u_0 的 Jacobi(雅可比)矩阵，又记作：

$$J(f(u_0))=\left.\frac{\partial(y_1,y_2,\ldots y_k)}{\partial(u_1,u_2,\ldots u_m)}\right|_{u_0}$$

到此，可知命令 jacobian 就是用来求解多元函数组的 Jacobi 矩阵。而它的使用格式为：

```
R = jacobian(w, v)
```

式中参数含义为：

w　以列向量形式表达的多元函数组。

V　以行向量形式表达的多元函数组。

jacobian 命令的具体使用请见下例。

例 14　试求例 13 中 u_1，u_2，u_3 的 Jacobi 矩阵

```
w=[u1; u2; u3];
v=[x, y];
R=jacobian(w, v)
```

结果为：

```
R =
[cos(y)+2*x+2*y, -x*sin(y)+2*x+2*y]
[sin(y)+y,  x*cos(y)+x]
[2*x-y, -x+2*y]
```

试求由 f1=ab/c+sin(az)/cosy，f2=aexp^{bxz}/y，f3= xb^{a}+yc^{z} 组成的函数组关于变量 a，b，c，x，y，z 的 Jacobi 矩阵。

```
syms a b c x y z
f1=a*b/c+sin(a*z)/cos(y);
f2=a*exp(b*x*z)/y;
f3=b^a*x+c^(z*y);
w=[f1; f2; f3];
v=[a b c x y z];
R=jacobian(w, v)
```

结果为：

```
R =
[b/c+cos(a*z)*z/cos(y), a/c, -a*b/c^2, 0, sin(a*z)/cos(y)^2*sin(y),
cos(a*z)*a/cos(y)]
[exp(b*x*z)/y , a*x*z*exp(b*x*z)/y , 0 , a*b*z*exp(b*x*z)/y ,
-a*exp(b*x*z)/y^2, a*b*x*exp(b*x*z)/y]
[b^a*log(b)*x, b^a*a/b*x, c^(z*y)*z*y/c, b^a, c^(z*y)*z*log(c),
c^(z*y)*y*log(c)]
```

能用 jacobian 命令求函数组的 Jacobi 矩阵，那么求多元复合函数的关于所有变量的导数的运算就容易了。与一元复合函数的求导相同，只要注意一下对各层函数的说明顺序，jacobian 命令就可解得相应导数的 Jacobi 矩阵。

例 15 试求以下多元复合函数关于 x，y，z 的导数矩阵

$$\begin{cases} f1=a/b/c; \\ f2=ab/c; \\ f3=b^a c+c^{ab}; \end{cases} \qquad \begin{cases} a=yxz; \\ b=zy+zx+xy; \\ c=x+y+z; \end{cases}$$

```
syms a b c x y z
a=y*x;
b=z*y+z*x;
c=x+y+z;
f1=a/b/c;
f2=a*b/c;
f3=b^a*c+c^(a*b);
w=[f1; f2; f3];
v=[x y z];
R=jacobian(w, v)
```

结果为：

```
R =
[
y/(z*y+x*z)/(x+y+z)-y*x/(z*y+x*z)^2/(x+y+z)*z-y*x/(z*y+x*z)/(x+y+z)^2,
x/(z*y+x*z)/(x+y+z)-y*x/(z*y+x*z)^2/(x+y+z)*z-y*x/(z*y+x*z)/(x+y+z)^2,
-y*x/(z*y+x*z)^2/(x+y+z)*(y+x)-y*x/(z*y+x*z)/(x+y+z)^2]
[y*(z*y+x*z)/(x+y+z)+y*x*z/(x+y+z)-y*x*(z*y+x*z)/(x+y+z)^2,
x*(z*y+x*z)/(x+y+z)+y*x*z/(x+y+z)-y*x*(z*y+x*z)/(x+y+z)^2,
```

```
y*x*(y+x)/(x+y+z)-y*x*(z*y+x*z)/(x+y+z)^2]
[ (z*y+x*z)^(y*x)*(y*log(z*y+x*z)+y*x*z/(z*y+x*z))*(x+y+z)+(z*y+x*z)^(y*
x)+(x+y+z)^(y*x*(z*y+x*z))*((y*(z*y+x*z)+y*x*z)*log(x+y+z)+y*x*(z*y+x*z)
/(x+y+z))                                                              ,
(z*y+x*z)^(y*x)*(x*log(z*y+x*z)+y*x*z/(z*y+x*z))*(x+y+z)+(z*y+x*z)^(y*x)
+(x+y+z)^(y*x*(z*y+x*z))*((x*(z*y+x*z)+y*x*z)*log(x+y+z)+y*x*(z*y+x*z)/(
x+y+z))                                                                ,
(z*y+x*z)^(y*x)*y*x*(y+x)/(z*y+x*z)*(x+y+z)+(z*y+x*z)^(y*x)+(x+y+z)^(y*x
*(z*y+x*z))*(y*x*(y+x)*log(x+y+z)+y*x*(z*y+x*z)/(x+y+z))]
```

以上就是多元复合函数的求导方法，关于隐函数的求导方法读者可根据隐函数求导实践应用上面的方法很容易完成。由于 Jacobi 矩阵在高等数学中的作用很大，在很多方面均有应用，应仔细练习。

7.2 空间曲线积分

7.2.1 空间曲线曲面

空间曲线和曲面在数学分析中经常用到，特别是对其切线、法平面等特性的应用。下面介绍这方面的内容。

- 曲线

在 R2 中的曲线一般由参数方程描述，其形式为：

```
x=x(t),
y=y(t),          (a≤t≤b)
z=z(t).
```

根据切线的定义，可得曲线上任一点 M(x，y，z)的切线方向为：

```
v=(x'; y'; z')=jacobian([x, y, z], t)
```

而且其过 M(x0，y0，z0)点的切线方程 F 为：

```
F=-[x; y; z]+[x0; y0; z0 ]+v0*(t-t0)=0
```

其过 M 点的法平面方程 G 为：

```
G=[x-x0; y-y0; z-z0]'*v0 =0
```

例 16　试求曲线 L 在 M 点的切线方向 v0，切线方程 F 及法平面方程 G

曲线 L 的参数方程为：

```
x=2*sin(t),
y=2*cos(t),       (a>0; b>0)
z=4*t
```

点 M 为 t=pi/4。

```
syms t x y z
x0=2*sin(t);
y0=2*cos(t);
z0=4*t;
w0=[x0, y0, z0];
v0=jacobian(w0, t);
t=pi/4;
v0=subs(v0)
t0=t;
syms t
F=[x; y; z]-[x0; y0; z0]-v0*(t-t0)
G=([x; y; z]-[x0; y0; z0]).'*v0
```

结果为：

```
v0 =
       1.4142
      -1.4142
            4
F =
[ x-2*sin(t)-(t-1/4*pi)*2^(1/2)]
[ y-2*cos(t)+(t-1/4*pi)*2^(1/2)]
[                     z-8*t+pi]
G =
(x-2*sin(t))*2^(1/2)-(y-2*cos(t))*2^(1/2)+4*z-16*t
```

- 空间曲面

空间曲面可以表示为：

z=f(x，y) ((x，y)∈D∈R^2)时，其上 M 点的切平面方程为：

F=(z-z_0)-(diff(f，x)*(x-x_0)+diff(f，y)*(y-y_0))=0

例 17 试求 $z=x^2+y^2$ 在点(3，4，9)处的切平面方程 F

```
syms x y z
x0=3;
y0=4;
z0=5;
f=x^2+y^2;
F=(z-z0 )-(diff(f, x)*(x-x0)+diff(f, y)*(y-y0))
```

结果为：

```
F =
z-5-2*x*(x-3)-2*y*(y-4)
```

另外，曲面经常还表示为隐函数的形式：

```
F(x, y, z)=0
```

则在 M 点的切平面方程 G 为：

```
G=diff(f, x)*(x-x0)+diff(f, y)*(y-y0)+diff(f, z)*(z-z0)=jacobian(f, [x y z])*[x-x0; y-y0; z-z0]'
```

其法线方向向量 v 为：

```
v=jacobian(f, [x y z])
```

其法线方程 V 为：

```
V=-[x; y; z]+[x0; y0; z0]+[diff(f, x); diff(f, y); diff(f, z)]|x=x0
 =-[x; y; z]+[x0; y0; z0]+v
```

例 18　试求曲面 S 上点 M_0(3，4，5)处的切平面方程 G，法线方向向量 V0 和法线方程 V

曲面 S 为：

```
F=x^2/3+y^2/4+z^2/5-10=0
syms x y z
x0=3;
y0=4;
z0=5;
v=jacobian(F, [x;  y;  z])
v0=subs(v, x, x0);
v0=subs(v0, y, y0);
v0=subs(v0, z, z0)
F=x^2/3+y^2/4+z^2/5-10;
G=v0*[x-x0; y-y0; z-z0];
V=-[x; y; z]+[x0; y0; z0]+v0'*t;
```

结果为：

```
v =
[ 2/3*x,  1/2*y,  2/5*z]
v0 =
     2     2     2
G =
2*x-24+2*y+2*z
V =
[ -x+3+2*t]
[ -y+4+2*t]
[ -z+5+2*t]
```

- 曲线的隐函数方程

空间曲线通常还以两平面的相交线的形式表达出来。如曲线 L 由下面的方程确定：

```
F(x, y, z)=0;
G(x, y, z)=0.
```

则由高等数学中相关定理确定，点 M 处的切线方向的矢量为：

```
v=jacobian(F, [x, y, z])'*jacobian(G, [x, y, z])
```

例 19 试求球面 R 与曲面 S 的交线的沿切线方向的单位矢量

球面 R： $R=x^2+y^2+z^2-R^2$；

曲面 S： $S=x^2+y^2+a^2z^2$

```
syms x y z R a
R=x^2+y^2+z^2-R^2;
S=x^2+y^2+a^2*z^2;
v=jacobian(F, [x, y, z]).'*jacobian(G, [x, y, z])
```

结果为：

```
v =
[ 4/3*x,  4/3*x,  4/3*x]
[     y,      y,      y]
[ 4/5*z,  4/5*z,  4/5*z]
```

7.2.2 第二型曲线积分

第二型曲线积分就是一个向量函数沿有向曲线的积分。它最普通的应用之一就是在物理中计算力 F 在路径 S 上作的功。由于是向量积分，计算量增大了很多，容易出错，下面介绍在 MatLab 中如何计算第二型曲线积分。

通常情况下，对曲线积分的积分路径 L 的描述为参数方程，如：

```
x=x(t), y=y(t), z=z(t)        (a≤t≤b)
```

对于曲线 L 上任意一点 M(x(t), y(t), z(t))，L 在 M 的正向单位切向量为：

```
T(M)=(x'(t)*i+y'(t)*j+z'(t)*k)/(x'(t)^2+y'(t)^2+z'(t)^2)^0.5
```

L 在 M 的有向弧微分为：

```
dL=T(M)dl=dl*(x'(t)*i+y'(t)*j+z'(t)*k)/(x'(t)^2+y'(t)^2+z'(t)^2)^0.5
  =x'(t)dt*i+y'(t)dt*j+z'(t)dt*k
  =dx*i+dy*j+dz*k
```

有了曲线上的有向弧微分，则一个向量函数 F(x, y, z)沿曲线 L 前进时的积分元为：

```
dw=F(x, y, z)*dL=F(x, y, z)*T(x, y, z)*dl
```

而在整条曲线上的积分则为：

$$W=\int_L F(x,y,z)\cdot\Gamma(x,y,z)\cdot dl$$

$$=\int_L F(x,y,z)\cdot(x'(t)dt\cdot i+y'(t)dt\cdot j+z'(t)dt\cdot k)$$

上式是计算第二型曲线积分所依据的公式。

在 MatLab 中，矢量的点积可以表达成一个行向量和一个列向量之间的矩阵相乘。因此，向量函数 F 和因子 T 均需表达成一个矩阵的形式：

```
F(x, y, z)=[F1(x, y, z); F2(x, y, z); F3(x, y, z)]
```

```
T(x, y, z)=[T1(x, y, z); T2(x, y, z); T3(x, y, z)]
```

F 与 T 以矩阵乘法相乘的结果 G，就是最后积分运算的待积函数，随后的工作，就是用 int 命令对函数 G 积分。

例 20　试计算向量函数 F(x，y，z)=yi−xj+(x+y+z)k 沿空间螺旋线 L 的积分，L 为：x=asint；y=acost；z=bt.(m t n)

```
syms x y z m n a b t
x=a*sin(t);
y=a*cos(t);
z=b*t;
F=[y; -x; (x+y+z)];
v=[x, y, z];
T=jacobian(v, t);
G=F.'*T
INTD=int(G, m, n)
```

结果为：

```
G =
a^2*cos(t)^2+a^2*sin(t)^2+(a*sin(t)+a*cos(t)+b*t)*b
INTD =
a^2*n-b*a*cos(n)+b*a*sin(n)+1/2*b^2*n^2-a^2*m+b*a*cos(m)-b*a*sin(m)-
1/2*b^2*m^2
```

7.2.3　GREEN GAUSS STOKES 公式

这 3 个公式是高等数学中非常重要的公式。它们的作用是将一个在某一区域 A 的难以进行的积分转化到另一区域 B 中，并对待积函数进行相应的转换，从而达到变换思路，简化积分计算量的目的。

这 3 个公式所包含的运算，不过是求导、矢量乘积、二重三重积分。这些运算的相关命令在前面的说明中已经介绍得相当详细，在此只是将这些命令加以组合。

另外，如在应用中使用这些公式的目的只是为了简化运算，可按照第二型曲线积分直接计算，对于繁琐的求导、积分运算，可由 MatLab 来完成。

7.3　LALACE FOURIER AND Z 的变换及逆变换

7.3.1　LALACE FOURIER AND Z 的符号变换及逆变换

傅里叶、拉普拉司、Z 变换及其逆变换。

在 MatLab 中，符号运算工具箱提供了对符号代数表达式进行 Fourier 变换、Laplace 变换和 Z 变换的专用命令。这些命令是非常实用的。

- Fourier 变换及其逆变换。

将函数 f(x)进行 Fourier 变换 $f = f(x) \Rightarrow F = F(w)$ 的计算公式为：

$$F(w) = \int_{-\infty}^{+\infty} f(x)e^{-iwx}dx$$

计算命令为 fourier，其使用格式为：

```
F = fourier(f)
F = fourier(f, v)
F = fourier(f, u, v)
```

第 1 种使用格式的作用是将关于变量 x 的表达式 f(x)进行变换得结果 F(w)。但当表达式是关于变量 w 即 f(w)时，变换结果为 F=F(t)。如式中没有这 2 个变量时，将对其默认变量进行变换。

第 2 种格式指定了变换结果是变量 v 的函数，表现在计算过程中则公式变为：

$$F(v) = \int_{-\infty}^{+\infty} f(x)e^{-ivx}dx$$

第 3 种格式在第 2 种格式的基础上进一步确定了要对函数表达式 f 作关于变量 u 的 fourier 变换。表现在计算过程中则公式变为：

$$F(v) = \int_{-\infty}^{+\infty} f(u)e^{-ivu}du$$

例 21

```
syms a b x
f1=x*exp(-abs(x));
f2=exp(-a^2);
F1=fourier(f1)
F2=fourier(f2, a, b)
```

结果为：

```
F1 =
-4*i/(1+w^2)^2*w
F2 =
pi^(1/2)*exp(-1/4*b^2)
```

将函数 F(x)进行 Fourier 逆变换 $F = F(w) \Rightarrow f = f(x)$ 的计算公式为：

$$f(x) = \frac{1}{2\pi}\int_{-\infty}^{+\infty} F(w)e^{-iwx}dw$$

计算命令为 ifourier，其使用格式为：

```
f = ifourier(F)
f = ifourier(F, u)
f = ifourier(F, v, u)
```

式中参数 F 为待进行逆变换的代数表达式，u，v 则与 fourier 命令中的用法完全一样。以第 3 种格式为例，它的作用是将表达式 F 进行关于变量 v 的逆傅里叶变换，并将结果表

示为变量 u 的函数。

例 22

```
syms w b a x
F1=exp(-w^2/(a^2));
F2=pi^(1/2)*exp(-1/4*b^2);
f1=ifourier(F1)
f2=ifourier(F2, b, a)
```

结果为：

```
f1 =
1/2*(pi*a^2)^(1/2)*exp(-1/4*x^2*a^2)/pi
f2 =
3991211251234741/4503599627370496*4^(1/2)/pi^(1/2)*exp(-a^2)
```

函数 F2 的逆变换结果为式 f2，但与前面 fourier 命令中的例子好像不一样，计算一下知：

```
A=3991211251234741/4503599627370496*4^(1/2)/pi^(1/2)
```

结果为：

```
A=1
```

- Laplace 变换及其逆变换

将函数 f(x)进行 Laplace 变换 $F = F(t) \Rightarrow L = L(s)$ 的计算公式为：

$$L(s) = \int_0^\infty F(t)e^{-st}dt$$

计算命令为 laplace，其使用格式如下：

```
laplace(F)
laplace(F, t)
fourier(F, w, z)
```

式中参数 F 为待进行 Laplace 变换的代数表达式。

第 1 种格式中，要求 F 为变量 t 的函数，命令 laplace(F)将按上面的标准，以 Laplace 变换的公式运算。如式中没有这个变量时，将对其默认变量进行变换。

第 2 种格式中，命令要求 F 以变量 x 为积分变量，进行如下形式的运算：

$$L(t) = \int_0^\infty F(x)e^{-tx}dx$$

结果自然就是变量 t 的函数。

第 3 种格式要求 F 以变量 w 为积分变量，进行如下运算：

$$L(z) = \int_0^\infty F(w)e^{-zw}dw$$

结果为 z 的函数。

例 23

```
syms x t v
F=sin(x*t+2*t);
L1=laplace(F)
L2=laplace(F, x, v)
```

结果为：

```
L1 =
(x+2)/(s^2+(x+2)^2)
L2 =
(v*sin(2*t)+t*cos(2*t))/(v^2+t^2)
```

将函数 F(x)进行 Laplace 逆变换 $L = L(s) \Rightarrow F = F(t)$ 的计算公式为：

$$F(t) = \int_{c-i\infty}^{c+i\infty} L(s)e^{st}ds$$

计算命令是 ilaplace，其使用格式为：

```
F=ilaplace(L)
F=ilaplace(L, y)
F=ilaplace(L, y, x)
```

式中 L 为待进行逆变换的表达式。

第 1 种格式的作用是对表达式 L 中的变量 s 进行积分，逆变换结果将是变量 t 的函数。如果表达式 L 中没有 s 而有变量 t，则命令将对 t 进行逆变换，其结果为变量 x 的函数。如式中没有这两个变量时，将对其默认变量进行变换。

第 2 种格式仍是对 L 中的变量 s 进行积分，而结果将为 y 的函数。

第 3 种格式将关于变量 y 进行逆变换，并按如下公式运算：

$$F(x) = \int_{c-i\infty}^{c+i\infty} L(y)e^{xy}dy$$

结果为 x 的函数。

例 24

```
syms x t v
F=sin(x*t+2*t);
L1=laplace(F)
L2=laplace(F, x, v)
```

结果为：

```
F1 =
(x+2)*(-i/(2*x+4)*cos((-x-2)*t)-1/(2*x+4)*sin((-x-2)*t)+i/(2*x+4)*co
s((x+2)*t)+1/(2*x+4)*sin((x+2)*t))
F2 =
sin(2*t)*cos(x*t)+cos(2*t)*sin(x*t)
```

可用 simple 命令核对结果是否与前面 Laplace 变换命令中的例子相同。

- Z 变换及其逆变换

将函数　f(x)进行 Z 变换 $f = f(n) \Rightarrow F = F(z)$ 的计算公式为：

$$F(z) = \sum_{0}^{\infty} \frac{f(n)}{z^n}$$

计算命令为 ztrans，其使用格式为：

```
F = ztrans(f)
F = ztrans(f, w)
F = ztrans(f, k, w)
```

式中 f 为待进行 Z 变换的表达式。

第 1 种格式当 f 的系统默认变量为 n 或 z 时，　ztrans(f)将对其进行 Z 变换，并且结果为 z 或 w 的函数。如式中没有这两个变量时，将对其默认变量进行变换。

第 2 种格式将按下式计算，其结果自然也就是 w 的函数。

$$F(w) = \sum_{0}^{\infty} \frac{f(n)}{w^n}$$

第 3 种格式将指定变量进行变换，其计算公式为：

$$F(w) = \sum_{0}^{\infty} \frac{f(k)}{w^k}$$

例 25　将函数 f=sin(xt+2t)分别进行关于 x 和 t 的 z 变换

```
syms x t y
f=sin(x*t+2*t);
F1=ztrans(f)
F2=ztrans(f, t, y)
```

结果为：

```
F1 =
z*(cos(2*t)*sin(t)-sin(2*t)*cos(t)+sin(2*t)*z)/(1-2*z*cos(t)+z^2)
F2 =
-sin(x+2)*y/(2*y*cos(x+2)-y^2-1)
```

将函数 F(x)进行 Z 逆变换的计算公式为：

$$f(n) = \frac{1}{2\pi i} \oint_{|z|=R} F(z) z^{n-1} dz, n = 1, 2, \ldots$$

计算命令是 iztrans，其使用格式为：

```
f = iztrans(F)
f = iztrans(F, k)
f = iztrans(F, w, k)
```

式中 F 为待进行 Z 的逆变换的表达式。

第 1 种格式当 F 的系统默认变量为 z 或 n 时，　iztrans(f)将对其进行 Z 逆变换，并且结果为 n 或 k 的函数。如式中没有这 2 个变量时，将对其默认变量进行变换。

第 2 种格式将按格式一中变量确认方法对该变量计算，其结果是 k 的函数。

第 3 种格式将指定变量 w 进行变换，其计算结果为 k 的函数。

例 6

```
syms z t y x n
F1=n*(n+1)/(n^2+2*n+1);
F2=x/(x^2-2*x*exp(z)+exp(2*z));
f1=iztrans(F1)
f2=iztrans(F2, x, n)
```

结果为：

```
f1 =
(-1)^k
f2 =
exp(z)^n*n/exp(z)
```

7.3.2 信号处理中的几个数值变换命令

下面将要介绍的数值变换命令，运行起来比起前面的符号运算命令要快得多。

- 线性调频 Z 变换命令 czt

该命令的使用格式为：

```
y = czt(x, m, w, a)
y = czt(x)
```

此命令用来计算信号 x 的线性调频 Z 变换。参数 m 用来设定变换长度，w，a 则用来定义 Z 变换所沿的螺旋周线。

- 离散余弦变换及其逆变换 dct/idct

该命令的使用格式为：

```
y = dct(x)
y = dct(x, n)
```

其中 x 为信号向量，当其长度(length(x))不等于 n 时，命令 dct(x，n)将通过补零或截断的办法整理 x。

此命令将按下面的公式进行变换：

$$y(k)=\sum_{n=1}^{N}w(n)x(n)\cos\frac{\pi(2n-1)(k-1)}{2N},k=1,\ldots,N$$

其中 $w(n)=N^{-1/2}$(当 n=1 时)，$w(n)=(2/N)^{1/2}$(当 2=N 时)。

由上式计算得的 y 将是信号 x 的归一化变换结果。

- 快速傅里叶变换及其逆变换 fft/ifft。

一维快速傅里叶变换命令 fft 的使用格式为：

```
y = fft(x)
y = fft(x, n)
```

式中 x 为信号向量，当 x 为矩阵时，此变换将分别对 x 的每一列进行。参数 n 的作用是在当 x 的长度不等于 n 时，通过补零或截断的办法整理 x。

此命令将按下面的公式进行变换：

$$X(k)=\sum_{j=1}^{N}x(j)\omega_N^{(j-1)(k-1)}$$

一维快速傅里叶逆变换命令 ifft 的使用格式为：

```
y = ifft(x)
y = ifft(x, n)
```

式中参数含义同前。此命令将按下面的公式进行变换：

$$X(j)=\frac{1}{N}\sum_{k=1}^{N}X(k)\omega-_N^{(j-1)(k-1)}$$

- 二维快速傅里叶变换及其逆变换 fft2/ifft2

二维快速傅里叶变换命令 fft2 的使用格式为：

```
Y = fft2(X)
Y = fft2(X, m, n)
```

式中 X 是信号矩阵。命令 Y=fft2(X)先对 X 的每一列进行一维快速傅里叶变换，然后对得到的中间结果(行向量)再进行一次一维快速傅里叶变换，从而得到二维快速傅里叶变换 fft2。命令 Y=fft2(X，m，n)则是在进行变换之前，先对 X 进行截断或补零，使之成为 m×n 阶矩阵。

二维快速傅里叶逆变换命令 ifft2 的使用格式为：

```
Y = ifft2(X)
Y = ifft2(X, m, n)
```

其中参数的含义与 fft2 命令中的含义相同。

7.4 常微分方程及方程组的求解

在 MatLab 中，符号运算工具箱提供了功能强大的求解常微分方程的符号运算命令 dsolve。常微分方程在 MatLab 中不能以如 y''+2y'=y 的形式出现，而要按如下规定重新表达：

符号 D 表示对变量的求导。Dy 表示对变量 y 求一阶导数，当需要求变量的 n 阶导数时，用 Dn 表示，D4y 表示对变量 y 求 4 阶导数。

由此，常微分方程 y''+2y'=y 在 MatLab 中，将写为'D2y+2Dy=y'。

7.4.1 求解无附加条件的常微分方程

对于无附加条件的常微分方程的解不同于偏微分方程，它的解就是该方程的通解，也

是下面其他条件下计算的依据。其使用格式为：

```
dsolve('diff_equation')
dsolve('diff_equation', 'var')
```

式中 diff_equation 为待解的常微分方程，第 1 种格式将以变量 t 为自变量进行求解，第 2 种格式则需定义自变量 var。

例 27

试解常微分方程 $x^2+y+(x-2y)y'=0$。

```
syms x t
diff_equ='x^2+y+(x-2*y)*Dy=0';
y1=dsolve(diff_equ)
y2=dsolve(diff_equ, 'x')
```

解得结果为：

```
y1 =
-1/2*(2*lambertw(-2*exp(-(2*x^2+t-C1)/x/(1+2*x))/x/(1+2*x))*x
+2*x+lambertw(-2*exp(-(2*x^2+t-C1)/x/(1+2*x))/x/(1+2*x)))*x
y2 =
[ 1/2*x+1/6*(9*x^2+12*x^3-36*C1)^(1/2)]
[ 1/2*x-1/6*(9*x^2+12*x^3-36*C1)^(1/2)]
```

式中 lambertw 为兰伯特 W 函数，且 y2 式中的未知系数 c1 标志解得结果为方程的通解。

7.4.2 求解带有初始条件的常微分方程

求解带有初始条件的常微分方程也很简单，一方面是由于只要提供通解和初始条件即可确定未知系数，另一方面是由于 MatLab 已经完成了上面的计算。它的使用格式为：

```
dsolve('diff_eqution', 'condition1, condition2, ...', 'var')
dsolve('diff_eqution', 'condition1', 'condition2', '...', 'var')
```

其中 condition 即为微分方程的初始条件。

例 28 试求常微分方程 $y'''-y''=x$ 的解

```
syms x y z a
diff_equ='D3y-D2y=x';
y=dsolve(diff_equ, 'y(1)=8', 'Dy(1)=7, D2y(2)=4', 'x')
```

解得结果为：

```
y =
-1/2*x^2-1/6*x^3+1/6-1/2*(-17*exp(2)+14*exp(1))/exp(2)*x
+7/exp(2)*exp(x)
```

7.4.3　求解常微分方程组

对若干个常微分方程联立求解的命令格式为：

```
dsolve('diff_equation1, diff_equation2, ...', 'var')
dsolve('diff_equation1', 'diff_equation2', 'var')
dsolve('diff_equation1, diff_equation2, ...', 'condition1,
condition2, ...', 'var')
dsolve('diff_equation1', 'diff_equation2', '...',
'condition1', 'condition2, ...', 'var')
```

前两种格式用于求解方程组的通解，后两种加上了初始条件，用于具体求解。

例 29

试求常微分方程组：

$$\begin{cases} f''+3g=\sin x \\ g'+f'=\cos x \end{cases}$$

的通解和在初始条件为 $f'(2)=0$，$f(3)=3$，$g(5)=1$ 的解。

```
diff_equ1='D2f+3*g=sin(x)';
diff_equ2='Dg+Df=cos(x)';
[gern_f, gern_g]=dsolve(diff_equ1, diff_equ2, 'x')
[f, g]=dsolve(diff_equ1, diff_equ2, 'Df(2)=0, f(0)=0, g(0)=0', 'x')
```

解得结果为：

```
gern_f =
C1-1/6*C2*3^(1/2)*exp(-3^(1/2)*x)+1/6*C2*3^(1/2)*exp(3^(1/2)*x)
-1/2*C3*exp(-3^(1/2)*x)-1/2*C3*exp(3^(1/2)*x)+C3+1/2*sin(x)

gern_g =
1/6*C2*3^(1/2)*exp(-3^(1/2)*x)-1/6*C2*3^(1/2)*exp(3^(1/2)*x)
+1/2*C3*exp(-3^(1/2)*x)+1/2*C3*exp(3^(1/2)*x)+1/2*sin(x)

f =
1/6*cos(2)*exp(2*3^(1/2))/(1+exp(4*3^(1/2)))*3^(1/2)*exp(-3^(1/2)*x)
-1/6*cos(2)*exp(2*3^(1/2))/(1+exp(4*3^(1/2)))*3^(1/2)
*exp(3^(1/2)*x)+1/2*sin(x)

g =
-1/6*cos(2)*exp(2*3^(1/2))/(1+exp(4*3^(1/2)))*3^(1/2)*exp(-3^(1/2)*x
)+1/6*cos(2)*exp(2*3^(1/2))/(1+exp(4*3^(1/2)))*3^(1/2)*exp(3^(1/2)*x)+1/
2*sin(x)
```

7.4.4　求解线性齐次常微分方程组

对于齐次线性微分方程组 $X'=AX$，有如下求解定理：

如果矩阵 A 具有 n 个线性无关的特征向量 v_1，v_2，…，v_n，它们对应的特征值分别为 λ_1，λ_2，…，λ_n(不必各不相同)，那么矩阵：

$$\Phi(t)=[e^{\lambda_1 t},\ e^{\lambda_2 t},\ \ldots,\ e^{\lambda_n t}] \quad -\infty<t<+\infty$$

是常系数线性微分方程组 X'=AX 的一个基解矩阵。

按此定理，可写出求解程序：

```
function y=dsolves(A)
syms t real
d=eig(A);
[v, n]=eig(A);
y=exp(d*t)'*v;
```

使用时，只需将矩阵 A 作为函数 dsolves 的参数 A 即可。

例 30

```
a=[3  5; -5  3];
X=dsolves(a)
```

解得结果为：

```
X =
[1/2*exp((3-5*i)*t)*2^(1/2)+1/2*i*exp((3+5*i)*t)*2^(1/2)         ,
1/2*exp((3-5*i)*t)*2^(1/2)-1/2*i*exp((3+5*i)*t)*2^(1/2)]
```

上面就是线性齐次微分方程组的解法，至于非齐次微分方程组的解法，由于只多了一个积分过程，这里就不再介绍。

另外，MatLab 除了可以解常微分方程，还可以数值求解偏微分方程。这涉及到了其专门的工具包 PDE，如感兴趣可以通过其帮助文件学习。

7.5 习　　题

- 多元函数的极限、微分、极值

(1) 多元函数的极限

① f=xyz/sinz　　在(1,2,0)处

② g=xz/sinx+xsiny　　在(3,4,7)处

(2) 多元函数的求导　附:求梯度的数值求法

① f=ytg(sinz)+xye^{xz}　　对变量 z 的偏导数、梯度

② g=[zsinxcosy；xy/z；e$^{zy/x}$]的 Jacobia 矩阵

(3) 显式复合函数微分求导

试求多元复合函数:

f1=uvsin(uv/w);
f2=exp(v/u−w^2);
f3=acos(u/v)asin(w);

$$\begin{cases} u=yxz; \\ v=zy/x+zx/y+xy/z; \\ w=x^2+y^2+z^2; \end{cases}$$

的关于 x，y，z 的导数矩阵。

(4) 隐函数的求导

① $F=x^3+\sin(yx)+z\cos z$ 求 dz/dx

② $G=a^2+b^3+c^4+1/abc$ 求 da/dc

(5) 多元函数的局部极值

① $z=x^3+y^3-3xy$

② $z=xy-3x^3-2y^2+10$

③ $z=xy-3/x-2/y$

将 7.1.4 节例中函数 f 的内容换为上面函数即可。

(6) 条件极值

① 在曲面 $F=4x^2+2y^2+3z^2-4xy-2yz=0$ 上求 $f=xy\sin z$ 的极值点。

② 在抛物线 $y^2=4x$ 上求一点,使其距离(2,7)最近。

③ 在椭球 $x^{2/3}+y^{1/2}+z^{2/5}=0$ 内放入一个立方体,求该立方体的最大体积。

④ 求两曲面 $z1=x^2+y^2$;$z2=x+y+10$ 的交线到原点的最近距离。

```
syms x y z lambda1 lambda2
f1=z-x^2+y^2;
```

- 空间曲线积分

(1) 空间曲线曲面

①求下列曲线在指定点处的切线及法平面。

a. $x=t-\sin t, y=1-\cos t, z=4\sin(t/4)$ 在($\pi/2-1,1,2\times2^{1/2}$)处

b. $x^2+y^2+z^2=8; z=x^2+y^2$ 在(2,2,72)处

② 求下列曲面在指定点处的法向量及切平面

a. $x^2+y^2+z^3=3$ 在(1,1,1)处

b. $z=x^2-y^2$ 在(4,1,15)处

(2) 第二型曲线积分

试求向量函数 $B=[x^2/y;\sin(yz);z]$在有向曲线 $x=t^2, y=\sin t+2, z=t$，$t\in[2，8]$上的积分。

- LALACE FOURIER AND Z 的变换及逆变换

(1) 完成函数 $f=1/x^{-|x|}+x$ 的 Fourier 变换及其结果的逆变换。

(2) 完成函数 $f=\sin(xt+2^t)+e^x+x$ 的 Laplace 变换及其结果的逆变换。

(3) 完成函数 $f=\sin(xt+x)+e^{x/t}$)的 Z 变换及其结果的逆变换。

- 常微分方程及方程组的求解

(1) 求解下列微分方程

① $y'=(x+y)(x-y)$

② $xy'=y\log(y/x)$ $y(10)=1$

③ $y'=-x\sin x/\cos y$ $y(2)=1$

(2) 求解下列微分方程组

① f'=f+3g,g'=f+4

② X'=AX 其中,

A=[2 44 8 3;3 4 2 9;9 23 8 43;92 4 1 4]

第 8 章　数据处理

在实际的试验和工程测量中，通过不同的方式测得的一些离散数据点，称作采样点。对这些数据进行利用之前，还要对其进行分析和处理，如剔除误差较大的或明显不正确的点，以提高数据的准确性。有时，由于条件的限制，不能通过现有的测量手段得到希望的数据量，则可以通过测量其他的量，并对这一量的测量数据进行运算，便可间接地得到所希望的数据量。假如要得到一个二维截面管道中的液体的速度分布，可以通过测量某些离散的点，并对其进行二维插值便可以得到截面上任一点的速度；而要得到整个截面上的流量，对这些离散的点在二维截面上进行数值积分则可。在本章中，主要介绍曲线拟合、数值插值、数值微商、数值积分等几个方面的内容。对于曲线拟合、数值微商和数值积分等方面的内容较复杂，为简单起见，仅限于讨论一维问题。

8.1　曲线拟合

在实际工程应用和实验中，要得到一条光滑的数据曲线或直线，而实际上却只能测得一些分散的数据点，因而在许多情况下，都需要利用这些不连续的数据点，运用最小二乘法等算法，利用多项式或其他的已知函数生成一个新的多项式或已知函数来对这些已知的数据点进行逼近。利用 MatLab 强大的绘图功能，可以将拟合而成的曲线绘制成图。实际中最常用的曲线拟合是多项式曲线拟合。

8.1.1　最小二乘法直线拟合

直线拟合从本质上说是曲线拟合，因而完全可以利用多项式拟合的方法进行计算。直线可以用一阶多项式来表示，它是特殊的多项式。在实际对实验数据进行分析中，最小二乘法进行直线拟合应用广泛，它在数据分析领域也是非常重要的，因而将它单独作为一部分。

下面给出一个利用一组数据[x_1 x_2 x_3 … x_n]和[y_1 y_2 y_3 … y_n]用最小二乘法求直线拟合系数的函数，其中 x 和 y 的数据个数必须相同，否则多余的数据将被忽略，并给出警告信息。

```
function yy=pline(x,y)
nx=length(x);ny=length(y);
if nx~=ny                                    %数据个数不相同，给出警告信息
  warning('The lengths of X and Y should be equal');
end
n=min(nx,ny);                                %数据个数太少，将会出错
if n<2
```

```
    error('the number of the DATA should be greater than 1');
    return;
end
```

执行过程如下：

```
x=x(1:n);y=y(1:n);
x=reshape(x,n,1);                         %生成列向量
y=reshape(y,n,1);
A=[x ones(n,1)];                          %连接矩阵 A
b=y;
B=A'*A;
b=A'*b;
yy=B\b;                                   %得到拟合系数
yy=yy';                                   %变成行向量
```

例 1 某实验中测得一组数据，其值如下：

```
x   1    2    3    4    5
y   1.3 1.8 2.2 2.9 3.5
```

已知 x 和 y 成线性关系，即 y=kx+b，求系数 k 和 b。

解：

```
x =[1    2    3    4    5];
y =[ 1.3000   1.8000   2.2000   2.9000   3.5000];
p=pline(x,y)
p =    0.5500    0.6900
```

可见，k 为 0.55，b 为 0.69，直线与数据点的关系用图来表示，如图 8.1 所示。

```
y1=polyval(p,x);
plot(x,y1)
hold on;
plot(x,y,'b*');
```

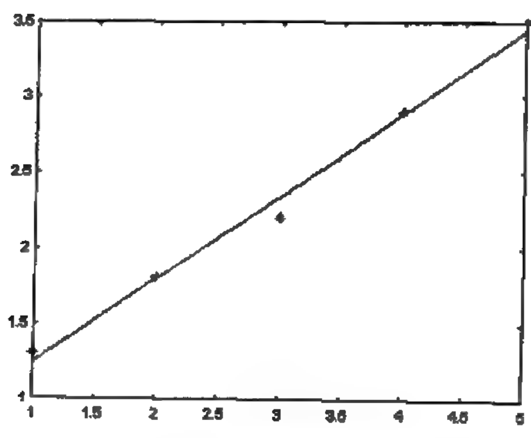



图 8.1 例 8.1 运行结果

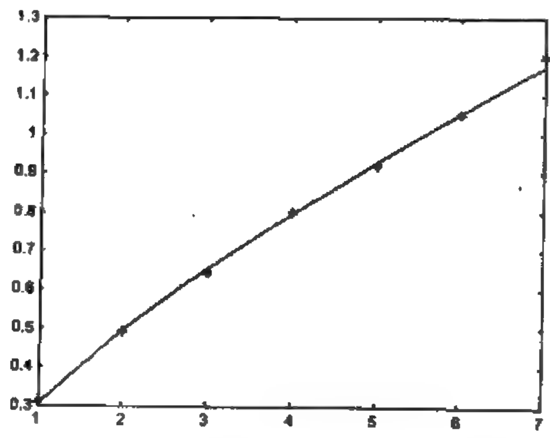


图 8.2 例 8.2 运行结果

例 2　在另一个实验中得到一组数据如下：

```
x =     1       2       3       4       5       6       7
y =     0.3101  0.4900  0.6400  0.8000  0.9200  1.0500  1.2000
```

而已知 x 和 y 满足关系式 $y=kx^n$，求参数 k 和 n。将方程两边取对数得：logy=logk+nlogx，可见 log(x)和 log(y)成线性关系。

将上面的数据放入变量 x 和 y 中，然后输入：

```
p=pline(log(x),log(y1))
p =   0.6909   -1.1851
exp(p(2))
ans =   0.3057
```

可知 n 的值为 0.6909，而 k 的值为 0.3057。

拟合所得的图形如图 8.2 所示。

8.1.2　多项式曲线拟合

多项式曲线拟合有最小二乘法、牛顿法、拉格朗日法、牛顿一格雷高里法等多种方法，其中最小二乘法是最普遍的多项式拟合方法，下面逐一介绍。

1. 最小二乘法

在 MatLab 中，用函数 polyfit()对一组数据进行定阶数的多项式拟合，其基本用法如下：

p = polyfit(x,y,n)用最小二乘法对输入的数据 x 和 y 用 n 阶多项式进行逼近，函数返回多项式的系数，为一个长度为 n+1 的向量，包含多项式的系数。

[p,s] = polyfit(x,y,n)函数不仅返回多项式的系数向量 p，还返回用函数 polyval()获得的误差分析报告，保存在结构体变量 S 中。

在多项式拟合中，如果 n 的值为 1，就相当于用最小二乘法进行直线拟合。

例 3　如上面的例 1 中，用 ployfit()可以求得：

```
 [p,s]=polyfit(x,y,1)
p =  0.5500    0.6900
s =    R: [2x2 double]
      df: 3
   normr: 0.1643
s.R
ans =-7.4162   -2.0226
        0   -0.9535
```

在用高阶多项式对某一函数进行曲线拟合时，并不是拟合出来的多项式与被拟合函数在整个区间上都能符合，polyfit()只能保证在输入数据 x 所能达到的区间上及其附近，求得的多项式可以最大限度地逼近原函数，而在其他的区间上，多项式并不一定能够很好地表示原函数，极有可能与原函数相差甚远的情况。在下例中，用 8 阶多项式来逼近正弦函数。

例 4 在此例中，用[0,3]区间上的数据来生成多项式，而在[0,6]的区间上画图，看看在哪些区域多项式能与函数很好地符合。

```
x=0:0.1:3;
y=sin(x);
p=polyfit(x,y,8)                          %用 x 和 y 在[0,3]上拟合 sin()
p =[ 0.0000   -0.0003   0.0002   0.0080   0.0003   -0.1668   0.0000
    1.0000   0.0000]                   %多项式的系数
x1=0:0.1:6;
y1=sin(x1);
y2=polyval(p,x1);
plot(x1,f,'k*',x1,y1,'k-')                    %在[0,6]区间上画图
```

画出的图形如图 8.3 所示。

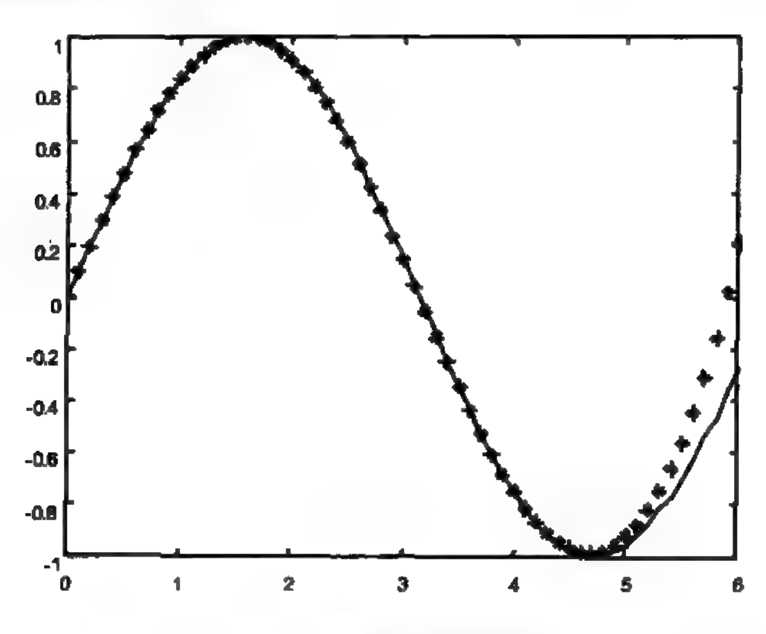


图 8.3 多项式的有效拟合区间

由图可知，多项式在区间[0,3]上与函数 sinx 符合得相当好，即使在区间以外的 4 处，也没问题，但是，在离拟合区间比较远的地方如 5 以后，差别就明显了，因而在拟合区间以外，用拟合所得的多项式来求某处的函数值不一定得到正确的结果。

2. 拉格朗日法

拉格朗日多项式拟合要求生成的多项式要过所有的数据点，如果有 n+1 个数据点，则拟合生成的多项式为 n 阶多项式。用拉格朗日法生成逼近多项式也是较常用的一种方法，它的计算有规律、简单、易于编程。但生成的多项式与用来插值的数据关系很大，如果数据有所改变，函数就要重新计算，如果数据很多，计算量则很大，因而在一定情况下很不方便。由于 MatLab 中没有给出用拉格朗日法进行多项式拟合的函数，这里，给出用此法进行多项式拟合的 M 文件的代码，此函数也含有用拉格朗日法进行插值的功能。

函数中，x 和 y 为拟合所用的数据，也是用来插值的原始数据，yy 为返回的拟合多项式，如果参数 z 存在，则 c 应该为 z 中的元素用拉格朗日插值法进行插值的结果。x 和 y 中的数的个数应该相同，否则会给出警告信息，但程序仍进行，多余的数据将被忽略。

```
function [yy,c]=lagrange(x,y,z)
nx=length(x);ny=length(y);
if nx~=ny
  warning('The lengths of X and Y should be equal');
end
n=min(nx,ny);
if n<2
  error('the number of the DATA should be greater than 1');
  return;
end
yy=0;
for i=1:n
  p=1.0;
  for j=1:n
    if i~=j
      if abs(x(i)-x(j))<eps
        error('the DATA is error!');
        return;
      end
     l1=[1 0-x(j)]/(x(i)-x(j));
     p=conv(p,l1);
     end
  end
  yy=pplus(yy,p*y(i));
end
if nargout==2
  c=polyval(yy,z);
end
```

例 5 利用拉格朗日法拟合下列的数据：

```
x =[    1       2       3        4        5]
y =[ 1.3000    1.8000    2.2000    2.9000    3.5000]
p=lagrange(x,y)
p =  -0.0333    0.4000   -1.6167    3.0500   -0.5000
polyval(p,x)
ans =  1.3000    1.8000    2.2000    2.9000    3.5000
```

多项式及各点的关系如图 8.4 所示，可见多项式过每个参加拟合的点。

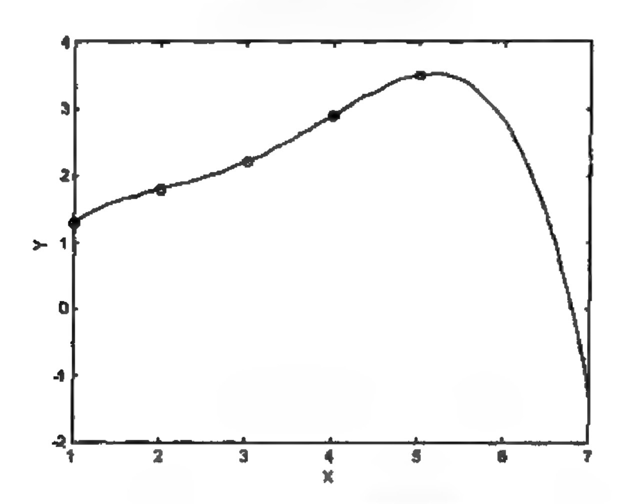


图8.4 拉格朗日法拟合数据点

3. 牛顿法

函数中，x和y为拟合所用的原始数据，yy为返回的拟合多项式，如果参数z存在，则c应该为z中的元素用拉格朗日插值法进行插值的结果。x和y中的数的个数应该相同，否则会给出警告信息，但程序仍进行，多余的数据将被忽略。牛顿法与拉格朗日法拟合所得的结果非常接近。关于牛顿法拟合的原理在此不作介绍，有兴趣请参阅有关文献，下面给出程序的代码：

```
function [yy,c]=newton(x,y,z)
nx=length(x);ny=length(y);
if nx~=ny
   warning('The lengths of X and Y should be equal');
end
n=min(nx,ny);
if n<2
   error('the number of the DATA should be greater than 1');
   return;
end
yy=0;
d=y;
for i=1:(n-1)
   for j=n:-1:(i+1)
      if abs(x(i)-x(j))<10*eps
         error('DATA input error!');
         return;
      end
      d(j)=(d(j-1)-d(j))/(x(j-i)-x(j));
   end
end
yy=d(n);
```

```
for i=(n-1):-1:1
   ym=conv(yy,[1; -x(i)]);
   yy=pplus(d(i),ym);
end
if nargout==2
   c=polyval(yy,z);
end
```

例 6 p=newton(x,y)

```
p =   -0.0333    0.4000   -1.6167    3.0500   -0.5000
polyval(p,x)
ans =    1.3000    1.8000    2.2000    2.9000    3.5000
```

说明多项式也过所有的数据点 。

例 7 用牛顿法在区间[1,3]上逼近正弦函数。

```
x=1:0.3:3;
y=sin(x);
p=newton(x,y)
p =  -0.0013    0.0120   -0.0053   -0.1638    0.0018    0.9970    0.0011
x1=0:0.1:6;
y1=sin(x1);
y2=polyval(p,x1);
plot(x1,y2,'ko',x1,y1,'k-')
```

图形如图 8.5 所示。

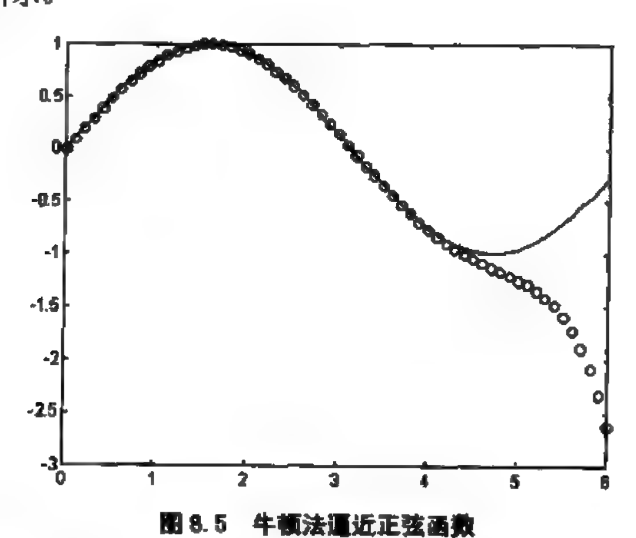


图 8.5 牛顿法逼近正弦函数

4. 牛顿—格雷高里法

如果有 n 个数据点，那么拟合产生的多项式的阶数也是 n-1 阶。所不同的是用牛顿—格雷高里法拟合所用数据的第 1 坐标必须是等间距的，而上面的拉格朗日法和牛顿法并不要求，虽然所举的例子中的数据的第 1 坐标都是等间隔的，但所有方法中，数据的第 1 坐

标都应按顺序排列。下面给出函数的程序代码，输入参数中，y 为数据的第 2 坐标所组成的向量，h 为数据的第 1 坐标的间隔，x0 为第 1 个数据的第 1 坐标。如果还要进行插值，则 xx 存放插值点的第 1 坐标组成的向量，如果 xx 存在，那么输出参数中，c 为相应的插值点的值，yy 为拟合产生的多项式的系数向量。

```
function[yy,c]=newtonfor(y,h,x0,xx)
n=length(y);
if n<2
   error('the number of the DATA should be greater than 1');
   return;
end
d=y;l=1;
for j=2:n
   for i=n:-1:j
         d(i)=d(i)-d(i-1);
      end
   end
   if abs(h)<eps
      error('h is too small');
      return;
   end
   s=[1 -x0]/h;
   yy=0;
   for j=0:(n-1)
      f=1;
      for i=0:(j-1)
         k=pplus(s,-i)/(j-i);
         f=conv(k,f);
      end
      yy=pplus(yy,d(j+1)*f);
   end
 if nargout==3
    c=polyval(yy,xx);
 end
```

例 8

```
p=newtonfor(y,1,1)
p =   -0.0333    0.4000   -1.6167    3.0500   -0.5000
```

可见，在用上面的 3 种方法对数据 x 和 y 进行多项式拟合时，所产生的多项式都是一样的，事实上，用 polyfit()进行拟合也可以得到同样的结果。

```
polyfit(x,y,4)
ans =  -0.0333    0.4000   -1.6167    3.0500   -0.5000
x=0.2:0.02:0.4;
y=humps(x);
p=newtonfor(y,0.02,0.2)
```

```
p =  1.0e+011 *
   -1.8582    5.5746   -7.4663    5.8779   -3.0119    1.0496   -0.2520
    0.0411   -0.0044    0.0003    0.0000
x1=0.15:0.02:0.4;
y1=humps(x1);
y2=polyval(p,x1);
plot(x1,y2,'ko',x1,y1,'k-')
```

所得图形如图 8.6 所示。

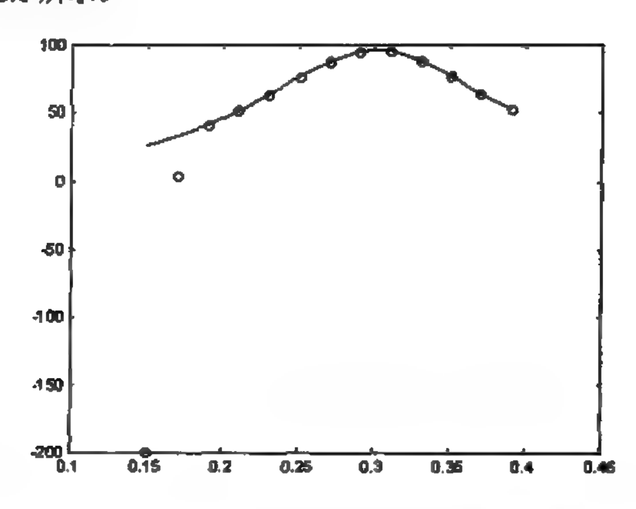


图 8.6　牛顿－格雷高里法拟合 humps 函数

5. 埃尔密特法

在实验所测得的数据中，既知道数据的第 1 坐标和函数值，也知道各点的一阶导数值，如果此时要用多项式进行曲线拟合，使参加拟合的各原始点的函数值和坐标要满足拟合所得的多项式，而且要使其一阶导数和第 1 坐标也要满足拟合多项式的微分多项式，则这些条件是非常必要的。下面给出埃尔密特拟合的函数及其说明。

在输入参数中，x0,y0 为参加拟合的原始点(x0(i),y(i))，y1(i)在此点的一阶导数值：xx 为参加插值的点的第 1 坐标组成的向量。输出参数中，yy 为拟合多项式的系数向量，c 为与 xx 对应的插值向量，只有当 xx 有值时 c 才有值。

```
function [yy,c]=aimet(x0,y0,y1,xx)
n=min(length(x0),min(length(y0),length(y1)));
if n~=length(x0)|n~=length(y0)|n~=length(y1)
  warning('the lengths of the arguments are not equals');
end
if n<2
  error('Not enough DATA,please check again.');
  return;
end
yy=0;
for i=1:n
```

```
      h=1;a=0;
      for j=1:n
         if j~=i
            if abs(x0(i)-x0(j))<eps
               error('DATA input error');
               return;
            end
            k=[1 -x0(j)]/(x0(i)-x0(j));
            kk=conv(k,k);
            h=conv(h,kk);
            a=1/(x0(i)-x0(j))+a;
         end
      end
      k=[-1 x0(i)]*(2*a*y0(i)-y1(i));
      kk=pplus(k,y0(i));
      kkk=conv(h,kk);
      yy=pplus(yy,kkk);
   end
   if nargin==4
      c=polyval(yy,xx);
   end
```

例 9 已知在某实验中测得的某质点的位移和速度随时间的变化如下，求质点的速度和位移随时间的变化曲线。

```
t =[    0     0.5000    1.0000    1.5000    2.0000    2.5000    3.0000]
y =[    0     0.4794    0.8415    0.9975    0.9093    0.5985    0.1411]
y1 =[ 1.0000    1.5000    2.0000    2.5000    3.0000    3.5000]
p=aimet(t,y,y1)
p = 1.0e+005 *
    0.0002   -0.0034    0.0295   -0.1477    0.4789   -1.0531    1.5978
   -1.6675    1.1689   -0.5222    0.1333   -0.0147    0.0000         0
pp=polyder(p)
pp = 1.0e+006 *
    0.0002   -0.0041    0.0324   -0.1477    0.4310   -0.8424    1.1185
   -1.0005    0.5845   -0.2089    0.0400   -0.0029    0.0000
polyval(pp,t)
ans = 1.0000    1.5000    2.0000    2.5000    3.0000    3.5000    4.0000
x=0:0.1:3;
y0=polyval(p,x);
yy=polyval(pp,x);
plot(x,y0,'k-',x,yy,'k--');
text(1,-40,'虚线为速度曲线')
text(1,-60,'实线为位移曲线')
hold on
plot(t,y0,'k*',t,y1,'ko')
xlabel('时间 t(秒)')
```

```
ylabel('速度(米/秒)和位移(米)')
```

绘得的图形如图 8.7 所示，其中虚线为速度曲线,实线为位移曲线。

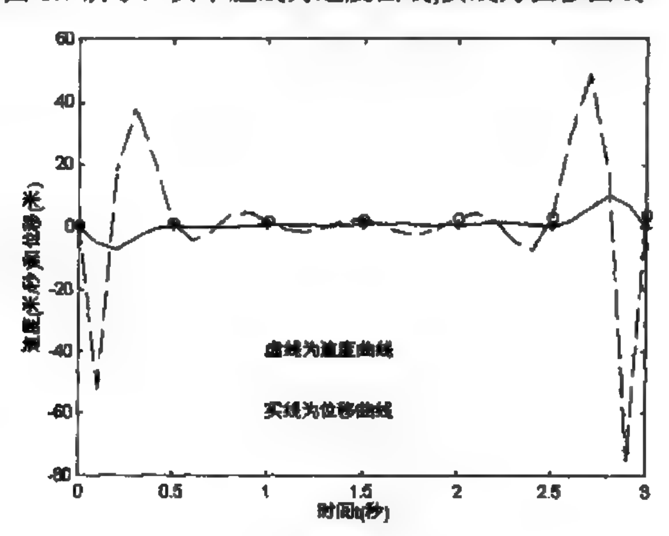


图 8.7 埃尔密特法拟合数据点

8.2 数 值 插 值

在工程测量和实验中，所得到的数据通常都是一些不连续的点。在流动、传热及燃烧等问题中运用数值计算进行求解时通常也是得到的一些不连续的点。如果要得到这些分散点以外的其他地方的数值，就必须运用这些已知的点进行插值。插值可以分为一维插值、二维插值和多维插值，按插值的方法可以分为线性插值、最近插值、最佳均方逼近插值等不同的插值形式。下面逐一介绍。

8.2.1 一维数值插值与查表

对于一维数据可以通过插值与查表来求得离散点之间的数据量。对于一维插值，按方法可以分为最近插值、线性插值、三次样条插值、三次插值，关于不同插值方法的插值可参阅有关方面的书籍。

1. interp1()函数插值

yi = interp1(x,Y,xi)函数返回的向量 yi 是向量 xi 根据数据 x 和 Y 的插值,求插值与查表类似。Interpl 对于 xi 中的元素，根据它们的位置，可以由 Y 的元素插值得到。如果 Y 是一个矩阵，那么插值对 Y 的每一列进行，函数返回一个 length(xi)×length(x)的矩阵 yi。如果 xi 中的元素不在 x 的范围内，则插值结果为 NaN。

yi = interp1(x,Y,xi,method)函数用 method 所指定的方法进行插值，method 的取值及插值方法如下：

```
'nearest'   最近插值
'linear'    线性插值
'spline'    三次样条插值
'cubic' 三次插值
```

对于上述所有的插值方法，都要求 x 的元素单调排列，如果 x 的元素为等间隔的，可以使用快速插值法，method 相应的取值为'*nearest'、'*linear'、'*spline'、'*cubic'。函数默认的插值方法为线性插值。

例 10 在某实验中要测某短时光束的发光强度随时间的变化，已知在实验中测得一组数据如下，t 为测数据的时间，y 为反映光束强度的指标，求 t=2.25 秒处的光的强度指标。

```
t = 0    0.5000   1.0000   1.5000   2.0000   2.5000   3.0000
y =0   0.4794   0.8415   0.9975   0.9093   0.5985   0.1411
```

用三次样条插值来求解：

```
interp1(x,y,2.25)
ans =  0.7539
```

通过绘图，可以看到插值后的函数与原始插值数据之间的位置关系，如图 8.8 所示。

```
x1=1:0.1:3;
y1=interp1(x,y,x1,'spline');
plot(x1,y1,'k-',x,y,'ko')
hold on
plot(2.25,interp1(x,y,2.25,'spline'),'k*')
```

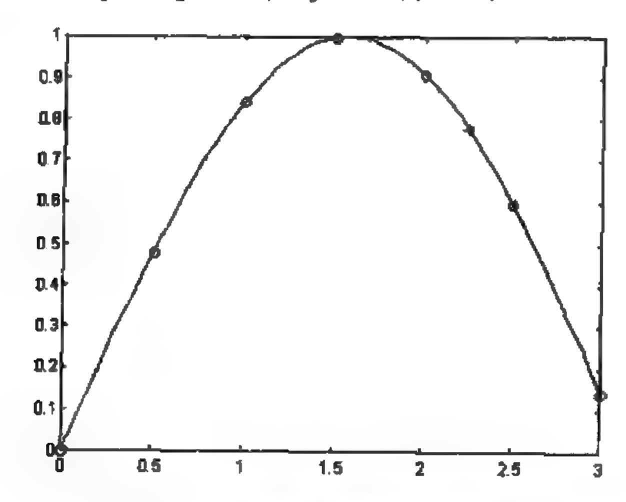


图 8.8 三次样条插值所得插值曲线

如果求 3.5 秒处的强度，则会出现下面的情况：

```
interp1(x,y,3.5)
ans =  NaN
```

2. 三次样条插值

yi = spline(x,y,xi)用三次样条插值法利用数据 x 和 y 在 xi 处进行插值，函数返回插值的

结果，x 为原始数据的第 1 坐标，y 为函数值。xi 可以为向量或矩阵。函数等同于 interp1(x,y,xi,'spline')。

pp = spline(x,y)得到用三次样条插值的分段多项式，可以用函数 ppval(pp,x)来计算多项式在 x 处的值。

例 11 spline(t,y,2.25)

```
ans =  0.7777
interp1(t,y,2.25,'spline')
ans =  0.7777
pp=spline(t,y);
ppval(pp,2.25)
ans =  0.7777
```

3. 一维表的查找

y=table1(tab,x)，从表 tab 中得到有若干线性插值列的表，tab 的第 1 列相当于数据的第 1 坐标 x，第 1 列数据必须单调，tab 的其余列为插值的函数值，按列进行插值，x 可以是向量，返回的变量 y 的列数是表 tab 的列数减 1，而 y 的行数是向量 x 的长度。x 中元素的值必须在 tab 的第 1 列的范围内，否则会出错。

例 12

```
tab =
         0         0    1.0000
    0.5000    0.4794    1.0000
    1.0000    0.8415    1.0000
    1.5000    0.9975    1.0000
    2.0000    0.9093    1.0000
    2.5000    0.5985    1.0000
    3.0000    0.1411    1.0000
yy=table1(tab,[2.25 1.2 3.5])
??? Error using ==> table1
x0 larger than all values in first column
yy=table1(tab,[2.25 1.2])
yy =0.7539    1.0000
    0.9039    1.0000
```

8.2.2 二维数值插值与查表

对于二维数据，也有插值与查表两种方法，在实际中，一些较复杂的问题通常都是二维问题，因此二维数据的插值应用既广泛又重要。如通过测量一个物理平面上一些离散点的温度，再通过二维插值就可以求得整个物理平面上的温度分布，并且可以利用 MatLab 强大的绘图功能将温度分布以图形方式输出。

1. interp2()二维插值

interp2()是 MatLab 自带的插值函数，有许多种形式。

ZI = interp2(X,Y,Z,XI,YI)，X 和 Y 为原始数据的第一和第二维坐标，Z 为函数值，函数返回在 XI 和 YI 所指定的位置插值所得的函数值。X 和 Y 必须为单调的向量或用单调的向量以 meshgrid()格式形成的网格格式，建议用函数 meshgrid()来生成比较方便。矩阵 X 和 Y 指定了数据 Z 中的元素的位置。如果 XI 和 YI 指定的插值点的位置超出了 X 和 Y 所指定的范围，则在相应的点上返回 NaN 值。XI 和 YI 可以为矩阵，且维数必须相同，返回值 ZI 为与 XI 和 YI 有相同大小的矩阵，Z(i,j)就是函数在点(X(i,j),Y(i,j))的插值结果。XI 和 YI 可以用向量 xi 和 yi 来代替，此时，MatLab 认为在点 meshgrid(xi,yi)处进行插值，即相当于依次执行以下两步：[XI,YI]=meshgrid(xi,yi)，interp2(X,Y,Z,XI,YI)。

ZI = interp2(Z,XI,YI)相当于依次执行下面的命令：

```
[m,n]=size(Z);
x=1:n;y=1:m;
[X,Y]=meshgrid(x,y);
interp2(X,Y,Z,XI,YI);
```

ZI = interp2(Z,ntimes)在 Z 的各点之间插入数据点以扩展 Z，依次执行 ntimes 次。默认为 1 次。

ZI = interp2(X,Y,Z,XI,YI,method)指定插值方法，method 的取值及相应的插值方法为：

```
'nearest'   最近插值
'linear'    双线性插值
'cubic'     三次插值
```

如果 X 和 Y 是等间距的，则可以使用快速插值法，其 method 的相应取值为'*linear', '*cubic','nearest'。

例 13

```
z =    [ 1 ·  2
          3    4]
z1=interp2(z)
z1 = 1.0000    1.5000    2.0000
    2.0000    2.5000    3.0000
    3.0000    3.5000    4.0000
z2=interp2(z1)
z2 =1.0000    1.2500    1.5000    1.7500    2.0000
    1.5000    1.7500    2.0000    2.2500    2.5000
    2.0000    2.2500    2.5000    2.7500    3.0000
    2.5000    2.7500    3.0000    3.2500    3.5000
    3.0000    3.2500    3.5000    3.7500    4.0000
interp2(z2,[1.25 1.88],[2.23 ])                  %此处作为向量处理
ans =    1.6775    1.8350
interp2(z2,[1.25 1.88],[2.23 2.7])               %此处作为矩阵处理而不是向量
ans =    1.6775    2.0700
```

下面利用二维插值函数对 peak 函数进行插值，所得图形如图 8.9 所示。

```
[X,Y] = meshgrid(-3:.25:3);
Z = peaks(X,Y);
```

```
[XI,YI] = meshgrid(-3:.125:3);
ZI = interp2(X,Y,Z,XI,YI);
mesh(X,Y,Z);
hold on;
mesh(XI,YI,ZI+15);
```

下面部分的图形为插值的结果，上面部分为原函数的图形。

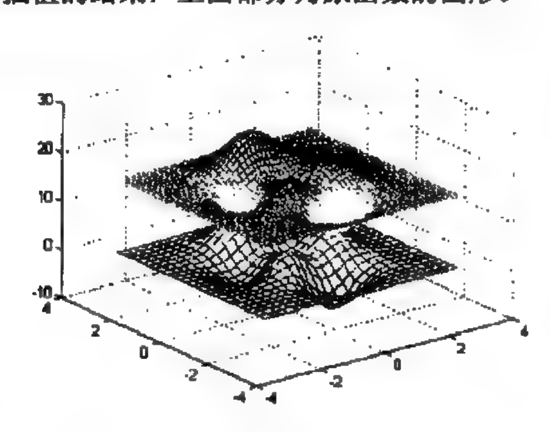


图8.9 二维插值函数对peak函数进行插值

2. 二维表的查找

Z = TABLE2(TAB,X0,Y0)从表TAB中得到一个线性插值交点，在表TAB的第1列中找X0，在第1行中找Y0，即矩阵TAB的第1列元素为原始数据的第1坐标，而第1行为其第2坐标，TAB(i,j)为点(TAB(i,1),TAB(1,j))处的值，TAB(1,1)会被忽略。TAB的第1行元素和第1列元素都应该是单调的。

例 14

```
z =  NaN     1.0000     2.0000     3.0000     4.0000     5.0000
    1.0000     1.0000     1.2500     1.5000     1.7500     2.0000
    2.0000     1.5000     1.7500     2.0000     2.2500     2.5000
    3.0000     2.0000     2.2500     2.5000     2.7500     3.0000
    4.0000     2.5000     2.7500     3.0000     3.2500     3.5000
         5.0000     3.0000     3.2500     3.5000     3.7500     4.0000

    table2(z,[2.23 ],[1.25 1.88])
ans =
1.6775    1.8350
table2(z,[2.23 2.7],[1.25 1.88])
ans = 1.6775    1.8350
   1.9125    2.0700
```

8.2.3 三维数值插值

在 MatLab 中，用函数 interp3()来计算三维函数的插值，其基本用法为：

```
VI = interp3(X,Y,Z,V,XI,YI,ZI)
VI = interp3(V,XI,YI,ZI)
VI = interp3(V,ntimes)
VI = interp3(...,method)
```

X，Y，Z 分别为第一、第二、第三维坐标，V 为函数值，XI、YI、ZI 为插值点的坐标，它们可以为向量或矩阵。设函数的基本用法与一维或二维插值函数基本相同，在此不作赘述。

8.2.4 最佳均方逼近

在实际的实验研究中，测出的数据非常多，用上面的多项式插值方法进行插值时多项式次数太高，并不理想。利用样条插值虽然可以使函数光滑且经过所有的数据点，但表达式较复杂，如果数据太多，则会占用大量的系统资源。对于实验数据本身而言，让插值的函数经过所有的数据点并无必要，因各个数据都存在着误差，而最佳均方逼近则较好地克服了以上不足之处。最佳均方逼近就是利用一系列已知函数来逼近未知的函数，只要知道参加所有的逼近的已知函数的参数即可。

下面给出最佳均方逼近的函数的原代码。

函数 evalf(f,x)中 x 是输入参数，用来执行以字符串 f 表示的函数，其中 f 为字符串，表示函数名，x 是数，返回函数的结果。

```
function y=evalf(f,x)
f=deblank(f);
ff=[f '(' num2str(x) ')'];
y=eval(ff);
```

例 15

```
evalf('sin',13)
ans =    0.4202
```

主函数 mini_product()用来产生最佳均方逼近的系数和插值的函数值。各参数的意义如下：f 为参加逼近的所有函数名组成的字符串矩阵，每个函数名占一行，注意每行的字符个数要相同，不同用空格补齐；向量 x 和 y 是参加逼近的原始数据的第 1 坐标和函数值，它们的长度必须相同，否则会出错；xx 是要进行插值的点的第 1 坐标组成的向量。返回参数中，a 为参加逼近的各函数的系数组成的向量，其长度与矩阵 f 的列的长度相同，即各行对应；e 为均方误差；yy 为与 xx 对应的插值点的函数值。

```
function [a,e,yy]=mini_product(f,x,y,xx)
m=size(f);m=m(1);
n=length(x);
b=zeros(m,m);c=zeros(m,1);
if n~=length(y)
```

```
        error('X and Y should be the same size.');
    end
    for j=1:m
        for k=1:m
            b(j,k)=0;
            for i=1:n
                b(j,k)=b(j,k)+evalf(f(j,:),x(i))*evalf(f(k,:),x(i));
            end
        end
        c(j)=0;
        for i=1:n
            c(j)=c(j)+evalf(f(j,:),x(i))*y(i);
        end
    end
    a=b\c;                                  %产生系数向量 a
    e=0;
    for i=1:n                               %计算均方误差 e
        ff=0;
        for j=1:m
            ff=ff+a(j)*evalf(f(j,:),x(i));
        end
        e=e+(y(i)-ff)*(y(i)-ff);
    end
    if nargin==3
        return;
    end
    yy=[];
    for i=1:m                               %计算 xx 的各元素对各逼近函数的函数值
        l=[];
        for j=1:length(xx)
            l=[l evalf(f(i,:),xx(j))];
        end
        yy=[yy l'];
    end
    yy=yy*a;                                %计算各插值点的函数值
    a=a';
```

例 16 对于下面的一组数据，用函数 y=x 和函数 y=1 进行逼近

```
x =[1     2     3     4     5];
y =[ 1.3000    1.8000    2.2000    2.9000    3.5000];
```

构造函数如下：

```
function y=ff0(x)
y=1;
function y=ff1(x)
y=x;
```

在工作空间中输入命令：

```
f=[ 'ff1'
`   'ff0']
mini_product(f,x,y)
ans =
    0.5500
    0.6900
```

可见上面的系数正是用最小二乘法进行直线拟合时所得到的直线的系数，即逼近的结果为 y=0.55x+0.69。

```
t = 0   0.5000   1.0000   1.5000   2.0000   2.5000   3.0000
y =0   0.4794   0.8415   0.9975   0.9093   0.5985   0.1411
```

用下列函数逼近上面的数据：sin(x),cos(x),humps(x),tan(x),x^3,x^2,x,1。

```
f1=['sin  '
'cos  '
'humps'
'ff3  '
'ff2  '
'ff1  '
'ff0  ']
xx=0:0.1:3;
[p,e,yy]=mini_product(f1,t,y,xx);
p =1.0000   0.0056   0.0000   -0.0008   0.0039   -0.0005   -0.0056
plot(t,y,'k*',xx,yy,'k-')
```

图形如下图 8.10 所示。

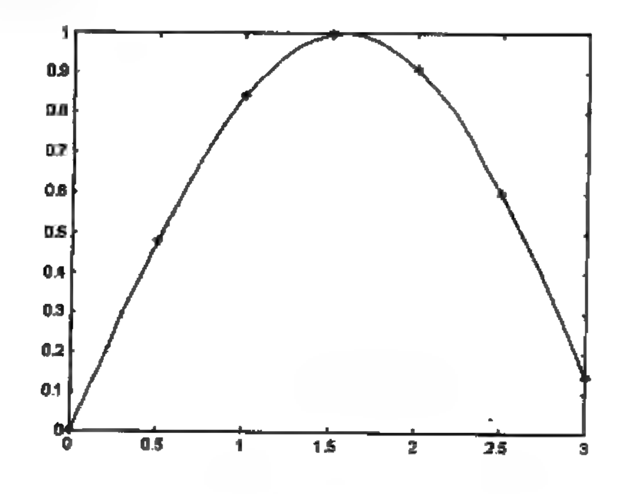


图 8.10 最佳均方逼近

8.3 数值微商

在实验或工程应用中，有时要根据已知的数据点，求某点的一阶或高阶导数，就要用到数值微分。数值微分的基本思路是先用逼近或拟合等方法将已知数据在一定范围内的近似函数求出，再用特定的方法对此近似函数进行求微分。通常有两种方法。下面逐一介绍。

8.3.1 多项式求导法求微分

已知函数某些节点的值，只要将用曲线拟合得到的多项式微分，再对微分后的多项式求值，便可以方便地求出在拟合范围内的任意一点的任意阶微分。曲线拟合给出的多项式，原则上是可以求任意阶导数，从而求出高阶导数的近似值，但随着求导阶数的增加，计算误差会逐渐增大，甚至与正确结果大相径庭，并且难以估计计算误差，因而本节讨论的只限于低阶数值微分。

例 17　用4阶多项式拟合函数sinx，并利用多项式的求导来求π/2的1阶和2阶导数。

```
x=0:0.3:3;
y=sin(x);
p=polyfit(x,y,4);                        %生成拟合多项式
pp=polyder(p);                           %求多项式的一阶微分
polyval(pp,pi/2)                         %求pi/2处的导数
ans =-3.1547e-004                        %精确解为0，误差为3.1547e-004
ppp=polyder(pp);                         %求多项式的2阶导数
polyval(ppp,pi/2)                        %求pi/2的2阶导数
ans = -0.9924                            %精确解为-1，误差为7.6e-3
```

可见随着导数阶数的增高，求得解的误差也增大，当然有时也会缩小，如求π/2处的3阶导数：

```
pppp=polyder(ppp);
polyval(pppp,pi/2)
ans = 0.0018                             %误差为1.8e-3
```

总之，高阶导数的误差是不确定的，难以估计误差的范围，这种不确定性是造成高阶导数的值的不准确性的根本原因。

为了完整，给出利用前面给出的多项式拟合函数来求n阶数值导数的函数。

```
function yn=deriv(x,y,z,n,flag)
if nargin==4
   flag=1;
end
if flag==1
   p=lagrange(x,y);
elseif flag==2
   p=newton(x,y);
```

```
else
   error('flag is out of index');
   return;
end
for i=1:n
   p=polyder(p);
end
yn=polyval(p,z);
```

可以修改上面的函数，使其能够利用多项式拟合函数 polyfit()和 newtonfor()等来求高阶微分。

下面利用此函数来求上例中 π/2 点的各阶导数。

例 18

```
deriv(x,y,pi/2,1)
ans = -3.2307e-010
deriv(x,y,pi/2,1,2)
ans = -9.0919e-011
deriv(x,y,pi/2,2,2)
ans =   -1.0000
deriv(x,y,pi/2,2,1)
ans =   -1.0000
deriv(x,y,pi/2,3,1)
ans =  5.5966e-009
deriv(x,y,pi/2,4,1)
ans =  1.0000
```

可见，用拉格朗日法和牛顿法拟合的多项式求高阶导数要比用 polyfit()函数拟合的多项式更加合理，精度更高。

8.3.2 中心差分法求微分

用中心差分法可以求解任意已知函数在某点的导数。对于所测得的实验数据，可以先用拟合和逼近法求出其近似函数的表达式，然后利用中心差分法进行求微分运算。

下面给出中心差分法求函数微分的代码。输入参数中，f 为字符串，是要求微分的函数的函数名；a 为求微分的点的坐标；h 为初始步长，初始步长的选择大小要合适，否则很可能得不到正确的结果，一般选择 0.1～0.01 之间的数值比较合适；tol 为计算精度，计算精度不要选择得太小，否则也有可能得不到正确的结果，一般选择 1e-4 左右；maxit 为计算的最大循环次数，此项是为了避免造成死循环而设立的。因为函数返回的最终结果与初始步长和计算精度都有关系，所以会有所差别。

```
function y=centdiff(f,a,h,tol,maxit)
ff=evalf(f,a);
if nargin<=4
  maxit=1000;
end
for i=1:maxit
```

```
    gg=(evalf(f,(a+h))-evalf(f,(a-h)))/(2*h);
    if abs(gg-ff)<tol
       y=gg;
       return;
    end
    h=h*0.5;
    ff=gg;
 end
 y=gg;
```

例 19

```
centdiff('cos',pi/2,0.01,1e-4)
ans = -1.0000
centdiff('cos',pi,0.01,1e-4)
ans = 7.3464e-006
centdiff('cos',pi/2,1,1e-4,10000)
ans =  -0.9984
centdiff('cos',pi/2,1,1e-6,10000)
ans = -1.0240
```

从以上结果中可见，微分所得结果与初始步长和求解精度有关。

```
centdiff('humps',0.3,1,1e-6,10000)       %求函数 humps()在 0.3 点的导数值
ans = 7.3728
```

8.4 数 值 积 分

在科学研究和工程应用中，除了要进行数据逼近外，有时还要求逼近曲线下面的面积，这就要求积分。在上一节中，已经分析了数值微分的运算，本节的求积分运算与之有一定相似之处，先利用多项式或其他函数对数据进行曲线拟合或逼近，得到数据的逼近函数，然后利用函数的数值积分函数来求某点的积分。在对已知函数求积分时，理论上可以利用牛顿一莱布尼兹公式求解，但在工程中却不实用，因为实际中遇到的大多数函数都不能找到其积分函数，有些函数的表达式非常复杂，用牛顿一莱布尼兹公式来求解会变得很繁琐，给计算精确带来不便。在对函数求积分时用数值积分就显得方便而精确，下面介绍几种求函数的数值积分的方法。

8.4.1 低阶法求数值积分

函数 quad()用 Simpson 递归法来求数值积分，其基本用法有：

q = quad('fun',a,b)　得到函数'fun'在区间[a，b]上的数值积分。

q = quad('fun',a,b,tol)　指定相对误差 tol，其默认值为 1.e-3。

q = quad('fun',a,b,tol,trace)　如果参数 trace 为非零值，则会以动态图形的形式实现积分的整个过程。

此函数并不能很好地处理异常的积分运算，例如，对函数 $x^{-1/2}$ 在[0，1]上求积分，因

```
    gg=(evalf(f,(a+h))-evalf(f,(a-h)))/(2*h);
    if abs(gg-ff)<tol
       y=gg;
       return;
    end
    h=h*0.5;
    ff=gg;
 end
 y=gg;
```

例 19

```
centdiff('cos',pi/2,0.01,1e-4)
ans = -1.0000
centdiff('cos',pi,0.01,1e-4)
ans = 7.3464e-006
centdiff('cos',pi/2,1,1e-4,10000)
ans =  -0.9984
centdiff('cos',pi/2,1,1e-6,10000)
ans = -1.0240
```

从以上结果中可见，微分所得结果与初始步长和求解精度有关。

```
centdiff('humps',0.3,1,1e-6,10000)        %求函数 humps()在 0.3 点的导数值
ans = 7.3728
```

8.4 数 值 积 分

在科学研究和工程应用中，除了要进行数据逼近外，有时还要求逼近曲线下面的面积，这就要求积分。在上一节中，已经分析了数值微分的运算，本节的求积分运算与之有一定相似之处，先利用多项式或其他函数对数据进行曲线拟合或逼近，得到数据的逼近函数，然后利用函数的数值积分函数来求某点的积分。在对已知函数求积分时，理论上可以利用牛顿—莱布尼兹公式求解，但在工程中却不实用，因为实际中遇到的大多数函数都不能找到其积分函数，有些函数的表达式非常复杂，用牛顿—莱布尼兹公式来求解会变得很繁琐，给计算精确带来不便。在对函数求积分时用数值积分就显得方便而精确，下面介绍几种求函数的数值积分的方法。

8.4.1 低阶法求数值积分

函数 quad()用 Simpson 递归法来求数值积分，其基本用法有：

q = quad('fun',a,b)　得到函数'fun'在区间[a，b]上的数值积分。

q = quad('fun',a,b,tol)　指定相对误差 tol，其默认值为 1.e-3。

q = quad('fun',a,b,tol,trace)　如果参数 trace 为非零值，则会以动态图形的形式实现积分的整个过程。

此函数并不能很好地处理异常的积分运算，例如，对函数 $x^{-1/2}$ 在[0，1]上求积分，因

```
 0.2190863625 0.1494513492 0.0666713443];
gg=0;
for i=1:n
   yy=(z(i)*(b-a)+a+b)/2;
   gg=gg+w(i)*evalf(f,yy);
end
y=gg*(b-a)/2;
```

例 22

```
gussled('sin',0,pi)
ans =  2.0000
gussled('ff3',0,1)
ans = 0.2334
```

2. 用似抛物线公式求高斯积分

f 为求积分的函数，[a, b]为积分区间，n 为分段数，分段越多，所求的积分越精确，但计算所需要的时间也越多，此函数求积分所用的时间就多。

```
function y=guass(f,a,b,n)
%n must be 2*m
h=(b-a)/n;
y=0;
for i=0:(1*n/2-1)
   y=y+h*(evalf(f,a+h*(1-1/sqrt(3))+2*i)...
      +evalf(f,a+h*(1+1/sqrt(3))+2*i));
end
```

例 23

```
guass('sin',0,pi,1000)
ans =  2.0000
guass('cos',0,pi,100)
ans = -3.0543e-006
guass('humps',0,3,1000)
ans = 23.9681
guass('humps',0,3,100)
ans = 23.9685
```

3. 复合辛普生法求积分

```
function y=comsimpson(f,a,b,n)
s=evalf(f,a)-evalf(f,b);
h=(b-a)/(2*n);x=a;
for i=1:2:(2*n-1)
   r=x+h;
   s=s+4*evalf(f,x);
   x=x+h;
   s=s+2*evalf(f,x);
```

```
end
s=s*h/3.0;
y=s;
comsimpson('sin',0,pi, 100)
ans =  2.0000
comsimpson('cos',0,pi, 100)
ans = 9.0803e-007
comsimpson('humps',0,3, 100)
ans = 23.9681
```

8.5 习　　题

(1) 已知 x=[1.2 1.4 1.8 2.1 2.4 2.6 3.0 3.3]，y=[4.85 5.2 5.6 6.2 6.5 7.0 7.5 8.0]，求对 x 和 y 进行直线拟合的拟合系数。

(2) 用最小二乘法及多项式拟合法分别求比较中的拟合系数，上题它们是否相同。

(3) 分别用 2、3、4、5 阶多项式来逼近[0,3]上的正弦函数 sin(x)，并做出拟合曲线及 sin(x)函数曲线图，了解多项式的逼近程度和有效拟合区间随多项式的阶数有何变换。

(4) 拉格朗日法、牛顿法、牛顿一格雷高里法、埃尔密特法对[0,3]上的正弦函数 sinx 进行多项式拟合，分析各拟合系数是否相同。

(5) 用 y=1，y=x，$y=x^2$，…等函数采用最佳均方逼近法逼近[0,3]上的正弦函数 sinx 并分析所得的系数与用多项式拟合法得到的系数是否相同。

(6) 已知 x=[0.1 0.8 1.3 1.9 2.5 3.1]， y=[1.2 1.6 2.7 2.0 1.3 0.5]，用不同方法求 x=2 点的插值，并分析所得结果有何不同。

(7) 用多项式求微分法和中心差分法求第 6 题中 x=2 处的点的 1 阶导数和 2 阶导数。

(8) 用不同的方法求题 6 中[0.1,3]区间上的数值积分值。

第 9 章　绘图及图像处理

MatLab 作为一个数学计算软件，之所以能受到众多使用者的青睐，除了在于它简单准确的数值计算功能和几十个功能强大的工具箱的支持之外，其出色的数据可视化和图像处理功能也起了非常重要的作用。

MatLab 的数据可视化和图像处理两大功能块，几乎满足了一般实际工程、科学计算中的所有图形图像需要。在数据的可视化部分，MatLab 可使用户计算所得的数据根据其不同情况转化成相应的图形。用户可以选择直角坐标、极坐标等不同的坐标系；它可以表现出平面曲线、空间曲线，绘制直方图、向量图、柱状图及空间网面图、空间表面图等。当初步完成计算结果的可视化图形后，MatLab 还可对图形作进一步加工，初级操作，如标注、添色、变换视角；中级操作，如控制色图、取局部视图、切片图；高级操作，如动画、句柄等。总之，这一系列命令和操作足以实实在在表达各种理想图形。

本章将详细介绍数据可视化的功能和图像处理的初级、中级功能，同时涉及了一些高级操作(这适于对 MatLab 有经验的用户)。通过下面的介绍，读者能很快掌握 MatLab 的基本图形操作，并能将其运用到实践中去。

9.1　窗　　口

图形窗口(Figure Window)是所有 MatLab 的图形输出的专用窗口。通过这个特殊窗口，可以自由查看和设置众多关于图形输出及表达的参数，并可获得高质量的图形打印文件。

9.1.1　图形输出窗口的创建与控制

当 MatLab 没有打开图形窗口时，如执行了一个绘图命令，该命令将自动创建一个图形窗口。如在命令执行以前，已经存在了若干个窗口，绘图命令将把图像输出到当前窗口中，并将原来存在于这个窗口的图像覆盖掉。

自己创建图形窗口的命令是 figure 命令。它有以下两种使用方法：

```
figure
figure(n)
```

图形窗口的名称是按照该窗口创建的时间顺序依次命名的：Figure No.1，　Figure No.2，... Figure No.n，因此命令 figure 将创建一个名为 Figure No.n+1 的新的空白图形窗口。不同的是，命令 figure(n)将创建一个名为 Figure No.n 的新空白图形窗口，不管前面的窗口 Figure No.n−1 是否存在。但如窗口 Figure No.n 已经存在，命令 figure(n)则将此窗口设置为当前窗口。

图形窗口作为各种图形图像操作命令的对象，具有很多参数，关于这些参数的具体设置方法将在相关部分进行介绍，这里先提供两个查阅这些参数和参数值的命令。

get(n)

命令 get(n)将返回关于图形窗口 Figure No.n 的所有图像参数的名称和当前值。

set(n)

命令 set(n)将返回关于图形窗口 Figure No.n 的所有图像参数的名称和其可能取的值。

9.1.2 多重子图窗口的创建

由于每个绘图命令在绘制数据图像时都会将前面已经绘得的图像覆盖掉，而用 hold 命令不能同时作好不同坐标尺寸下几个图像的显示，用 figure 命令再创建图形窗口又很难同时比较不同的数据绘得的图像。因此，可以在同一图形窗口内分割出几个子图，从而可在不同子图中绘制不同的数据图像。用于完成这个操作的命令是 subplot。它的使用格式为：

```
a=subplot(m, n, i)
```

此命令将当前窗口分割成 m×n 个子图，并将第 i 个子图作为当前视图，返回值 a 为当前视图的句柄值。其中每个子图都完全等同于一个完整的图形窗口，可在其中完成所有图形操作命令。这些图按行编号，即位于第 a 行 b 列处是其第(a-1)n+b 个子图。如果在执行 subplot 命令前系统没有创建任何图形窗口，则将创建出一个新的窗口并进行分割。

例 1

```
x=(-pi: 0.01: pi);
h1=subplot(2, 2, 1)
y1=sin(x);
plot(x, y1)
h2=subplot(2, 2, 2)
y2=cos(x);
plot(x, y2)
x=(-pi/2+0.1: 0.01: pi/2-0.1);
h3=subplot(2, 2, 3)
y3=tan(x);
plot(x, y3)
h4=subplot(2, 2, 4)
x=(0.1: 0.01: pi-0.1);
y4=1./tan(x);
plot(x, y4)
```

绘得的图形如图 9.1 所示。

结果为：

```
h1 =
          2.002685546875
h2 =
          4.003662109375
h3 =
         1.0054931640625
```

```
h4 =
          7.0013427734375
```

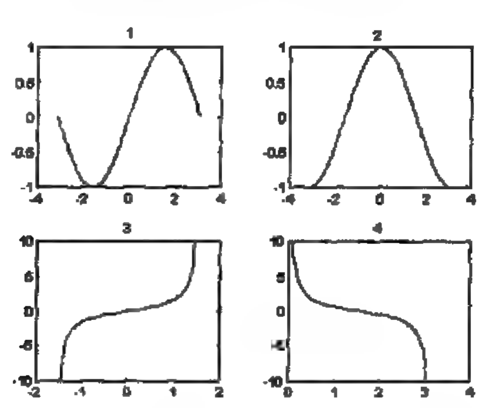


图 9.1　用 subplot 命令创建多个子窗口

9.2　二 维 绘 图

9.2.1　基本二维绘图命令

在第 3 章中，已介绍了 3 个基本的二维绘图命令。这些命令的使用方法和格式与下面介绍的其他二维绘图命令相似，因此这里有关部分的说明相应简单。

plot，fplot，ezplot 3 个命令都有一个局限，就是只能表现出函数值 y 随自变量 x 的线性变化，如遇到其中有在工程、科学计算中经常出现的指数变化，则前面 3 个命令就不能从图中直观地表现出来。

命令 loglog、semilogx、semilogy 则可以解决这个问题，它们的使用格式：loglog(…)，semilogx(…)，semilogy(…)完全与 plot 命令相同，但在坐标轴上有了变化：

loglog 命令将两个坐标轴分别变为 lnx，lny。

semilogx 命令只将横坐标变为 lnx。

semilogy 命令只将纵坐标变为 lny。

例 2

```
x=0: 0.1: 100;
y=exp(log(x).^2+4.*log(x)-3);
subplot(2, 2, 1)
plot(x, y)
subplot(2, 2, 2)
semilogx(x, y)
subplot(2, 2, 3)
semilogy(x, y)
subplot(2, 2, 4)
loglog(x, y)
```


绘得的结果如图 9.2 所示。

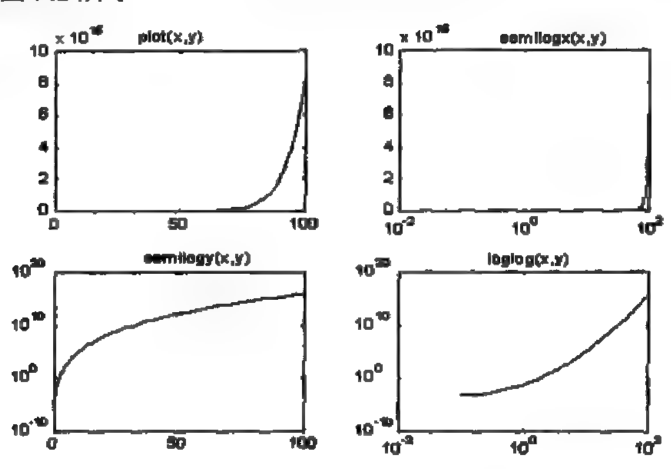


图 9.2　命令 plot, semilogx, semilogy, loglog 的比较

plotyy(x1，y1，x2，y2)命令的命令名中含有两个 y，它的作用是在相同的横坐标下使用两个纵坐标：(x1，y1)使用左侧的纵坐标，(x2，y2)使用右侧的纵坐标。其他的使用格式与 plot 命令一样。

例 3

```
x1=0: 0.01: 10;
x2=0: 0.001: 10;
y1=sin(x1);
y2=exp(abs(sin(x2)));
plotyy(x1, y1, x2, y2)
```

绘得结果如图 9.3 所示。

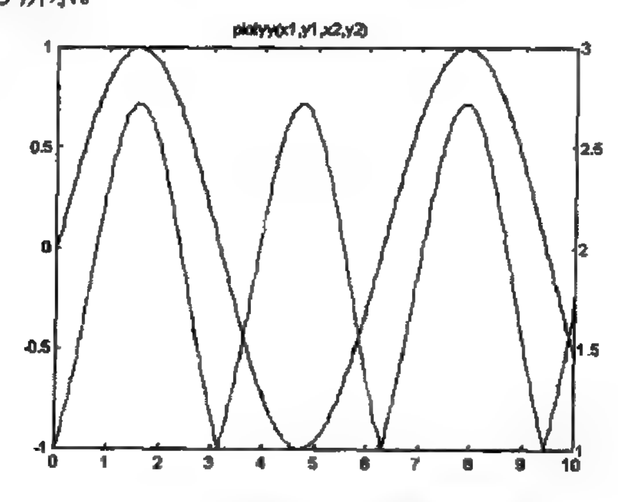


图 9.3　例 3 中命令 plotyy 的绘制结果

9.2.2 基本绘图控制参数：设置线型、线色和数据点

二维绘图命令 plot 为了适应各种绘图需要，提供了用于控制线型、线色和数据点的 3 组基本参数。它的使用格式如下：

```
plot(X, Y, 'color_linestyle_maker')
```

即用字符串'color_linestyle_maker'完成对上面 3 个参数的设置，其中具体用于控制各个参数的专用字符分别如表 9.1，9.2，9.3 所示。

表 9.1 颜色控制字符表

色彩字符	色 彩	RGB 值
y/yellow	黄色	1 1 0
m/magenta	洋红	1 0 1
c/cyan	青色	0 1 1
r/red	红色	1 0 0
g/green	绿色	0 1 0
b/blue	蓝色	0 0 1
w/white	白色	1 1 1
k/black	黑色	0 0 0

表 9.2 线型控制字符表

绘图字符	数 据 点	绘图字符	数 据 点
.	黑点	D	钻石形
o	小圆圈	V	三角形(向下)
x	差号	^	三角形(向上)
+	十字标号	<	三角形(向左)
*	星号	>	三角形(向右)
S	小方块	p	五角星
H	六角星		

表 9.3 数据点控制字符表

线型符号	线 型
-	实线
:	点线
-.	点划线
--	虚线

例 4

```
x=0: 0.25: 5;
y1=x.^0.1;
y2=x.^0.5;
y3=x.^0.8;
y4=x;
y5=x.^1.5;
y6=x.^2;
y7=cos(x);
y8=sin(x);
hold on
plot(x, y1, 'yo')
plot(x, y2, 'mx')
plot(x, y3, 'c+')
plot(x, y4, 'rs')
plot(x, y5, 'gh')
plot(x, y6, 'bd')
plot(x, y7, 'w<')
plot(x, y8, 'kp')
```

结果如图 9.4 所示。

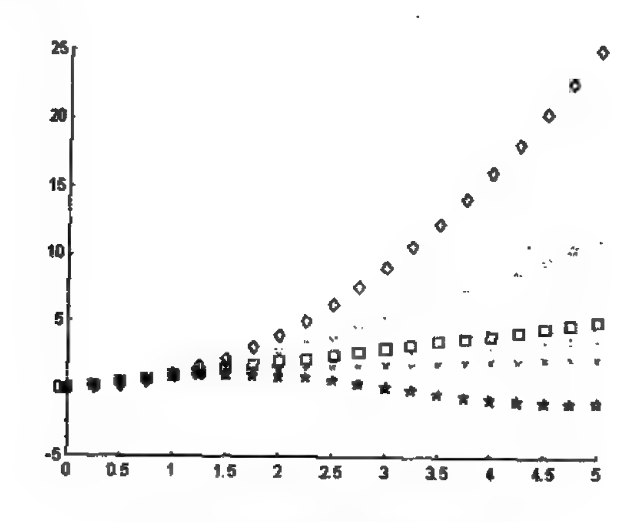


图 9.4 设置线型、线色和数据点

9.2.3 取点命令 ginput

该命令是 plot 命令的逆命令，它的作用是在二维平面图中记录下由鼠标所选点的坐标值。其使用格式为：

```
[x, y] = ginput(n)
[x, y] = ginput
[x, y, button] = ginput(...)
```

第 1 种格式中的参数 n 限制只能且必须在选择了 n 个点后才能结束该命令。第 2 种格式则无此限制，它可以无限制地选下去，当选够时，按 Enter 键结束命令。第 3 种格式的作用是在前两种格式的基础上加了一个返回值 button，这个值记录了在选取每个点时的相关信息。具体可参阅其帮助信息。

9.2.4 图形放大命令 zoom

zoom 命令是一个专门用来放大二维视图的命令"开关"。当需要查看绘得图形的某一微小部分时，只要将这个命令打开即可。.

zoom 命令的使用格式为：

```
zoom on
```

该命令使系统处于可放大状态。有两种方法来放大图形：

用鼠标单击需要放大的部分，可以将此处放大一倍，这一操作可连续进行，当右击时，图形将回到上一次的放大状态。

用鼠标拖出一个方框，系统将放大被框套住的部分。

zoom off

将系统转回到非放大状态，但前面放大所得的结果不会改变。

zoom out

将系统转回到非放大状态，并将图形恢复原状。

zoom reset

系统将记住此时图形的放大状态，当使用 zoom out 时，图形将不返回到原状，而是返回到 reset 时的放大状态。

zoom

该命令用于在 on 和 off 之间进行切换。

zoom xon

该命令只对 x 轴有放大作用。

zoom yon

该命令只对 y 轴有放大作用。

zoom(factor)

当 factor>1 时，系统将把图形放大 factor 倍；当 factor≤1 时，系统将把图形放大 1/factor 倍。

zoom(fig，option)

在选择非当前图形窗口 fig 后，将该窗口设为放大状态。option 就是前面介绍的 on，off，xon 等参数。

9.3 三 维 绘 图

9.3.1 三维基本绘图命令

在实际工程计算中，最常用的三维绘图是三维曲线图、三维网格图和三维曲面图 3 种.

基本类型。与此对应，MatLab 也提供了 3 个三维基本绘图命令(三维曲线命令 plot3、三维网格命令 mesh 和三维表面命令 surf)，下面分别介绍它们的具体使用方法。

- 三维曲线命令 plot3

plot3 命令与 plot 命令相同，均为 MatLab 的内部函数，它也是三维绘图的基本函数。其使用格式有：

```
plot3(x, y, z)
plot3(x, y, z, s)
plot3(x1, y1, z1, s1, ..., xn, yn, zn, sn)
```

当上式中的 x，y，z 为长度相同的向量时，plot 3 命令将绘得一条分别以向量 x，y，z 为 x，y，z 轴坐标值的空间曲线。

当上式中的 x，y，z 均为 m×n 的矩阵时，plot 3 命令将绘得 m 条曲线。其第 i 条空间曲线分别以 x，y，z 矩阵的第 i 列分量为 x，y，z 轴坐标值的空间曲线。

参数 s，s1，…，sn 就是在 plot 命令中的'color_linestyle_maker'控制字符，用来设置每条曲线的颜色、线型和数据点。

例 5 绘制螺旋线

```
t = 0: pi/50: 10*pi;
plot3(sin(t), cos(t), t);
```

绘制结果如图 9.5 所示。

例 6

```
[x, y]=meshgrid([-2: .1: 2]);
z=x.*exp(-x.^2-y.^2);
plot3(x, y, z)
```

结果如图 9.6 所示。

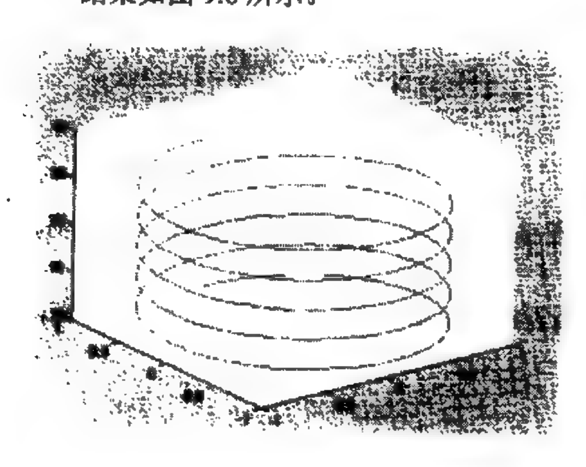


图 9.5 螺旋线图

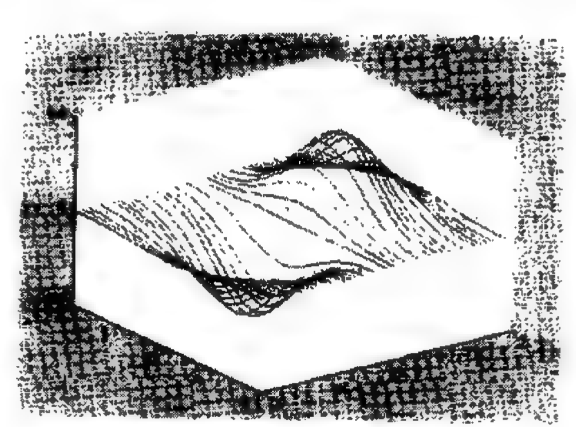


图 9.6 peaks 单线图

- 三维网格命令 mesh

此命令与 plot3 命令的区别在于要绘出的不是单根的曲线，而是一个在某一区间完整的曲面。其使用格式为：

```
mesh(z)
mesh(x, y, z)
```

式中 x，y 必须均为向量。若 x，y 的长度分别为 m 和 n，则 z 必须为 m×n 的矩阵。若参数中不提供 x，y，则将(i，j)作为 z 矩阵元素 z(i，j)的 x，y 轴坐标值。

例 7

```
[X, Y]=meshgrid(-12: .5: 12);
R=sqrt(X.^2+Y.^2)+eps;
Z=sin(R)./R;
mesh(Z)
```

结果如图 9.7 所示。

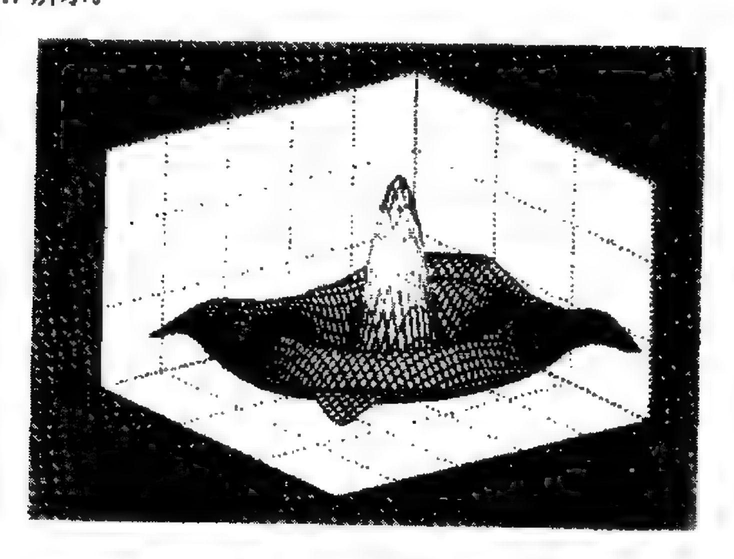


图 9.7 三维网格图

- 三维表面命令 surf

三维表面命令 surf 与 mesh 命令的用法及使用格式相同，不同之处在于绘得的图形是一个真正的曲面而不是用网格来近似表达的。

例 8

函数%peaks 为 MatLab 的内部函数。可用 type peaks 命令查看。

```
z=peaks;
surf(z)
```

结果为图 9.8 所示。

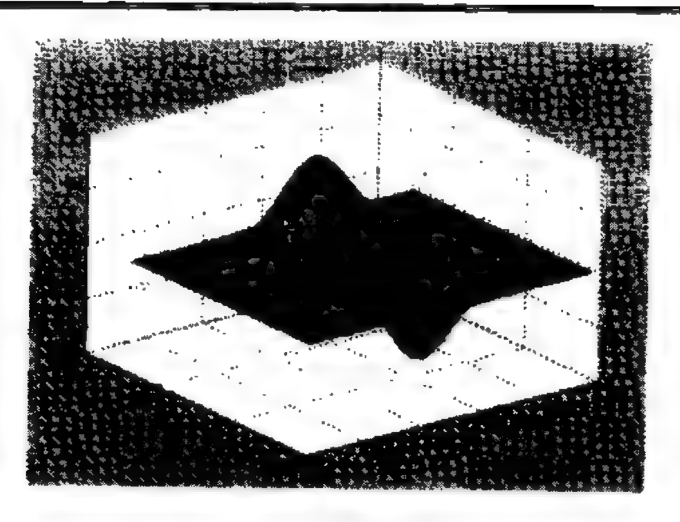


图 9.8 三维表面图

9.3.2 基本三维绘图命令的几个改进命令

在前面介绍的基本三维绘图命令 plot 3，mesh，surf 的功能只是最基本的三维绘图要求，下面是几个经常用到的命令，是由三维基本图形再加上一些特别处理，可以满足需要。

```
meshc, surfc
```

这两个命令用来在三维曲面图的下方绘出等高线。

例 9

```
meshc(peaks)
```

结果如图 9.9 所示。

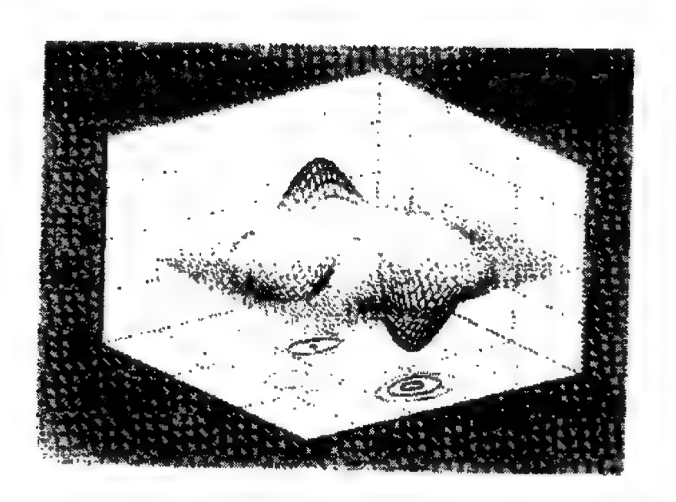


图 9.9 等高线图

```
meshz
```

此命令用来再加上一个参考平面。

例 10

```
meshz(peaks)
```

结果如图 9.10 所示。

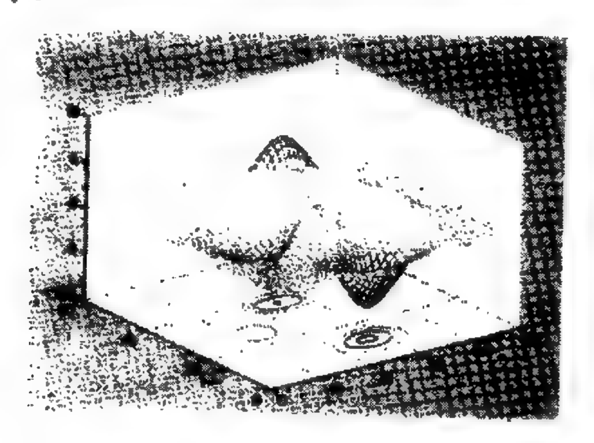


图 9.10　带有参考平面的网格图

pcolor

此命令用于绘制数据的伪彩图。其使用格式为：

pcolor(z)　以矩阵 z 的下标为横纵坐标绘制伪彩图。

pcolor(x，y，z)　以向量 x，y 为横纵坐标绘制伪彩图。

例 11

```
pcolor(peaks)
```

结果如图 9.11 所示。

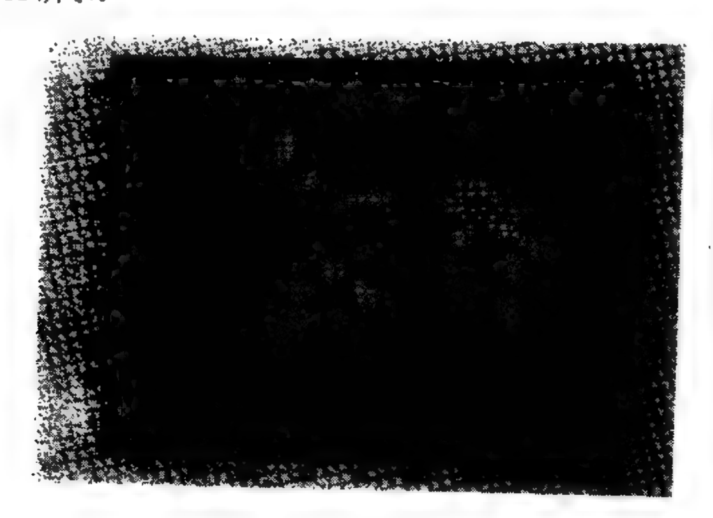


图 9.11　伪彩图

```
surfl
```

此命令用于绘制在控制光线的情况下的表面图。关于光线的控制方法将在 9.3.4 小节介绍。在默认情况下，光源位于从视线角度逆时针旋转 45 度的位置。

例 12

```
surfl(peaks)
```

结果如图 9.12 所示。

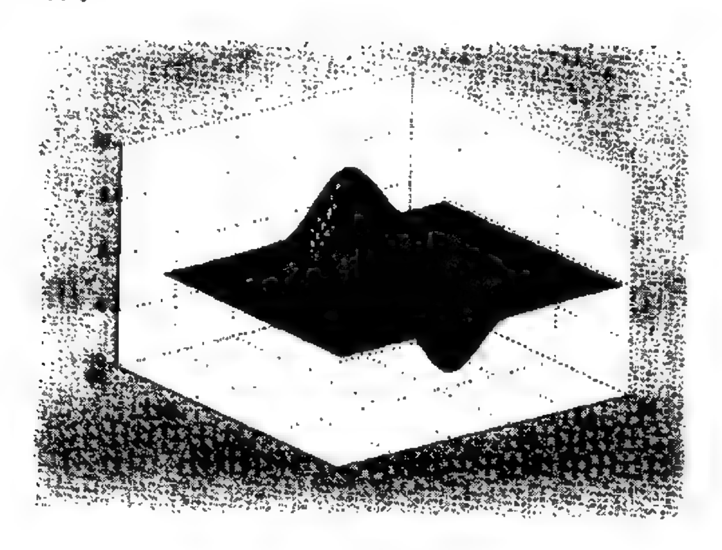


图 9.12 光线受控图

```
waterfall
```

此命令用于绘出类似瀑布流水形状的网线图。其使用格式同 mesh。

例 13

```
waterfall(peaks)
```

结果如图 9.13 所示。

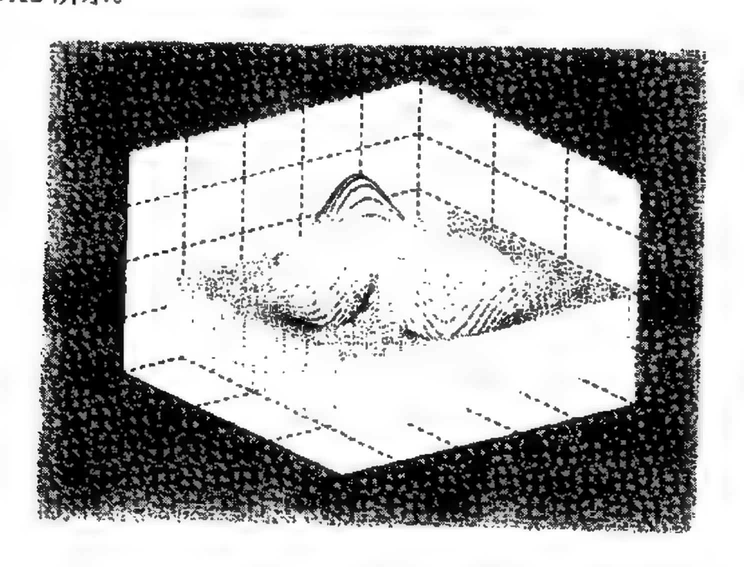


图 9.13 三维瀑布图

9.3.3　三维视图的可视效果控制

由于三维视图表现的是一个空间内的图形，因此从不同的位置和角度观察图形有不同的效果。另外，在复杂的三维图形中，经常会出现图形的某一部分被遮住的情况，这会对判断造成重要的影响，下面的两个命令会很好地解决这一问题。

设置观察三维图形的视角或视点 view。

view 命令用来控制三维图形的观察点和视角。其使用格式为：

1. view(az, el)

```
view([az, el])
view([x, y, z])
view(2)
view(3)
view(T)
[az, el] = view
T = view
```

各种使用格式的作用为：

前两种使用格式均是在球坐标系中设定视角的命令。其中 az 为方位角(azimuth)，el 为俯视角(elevation)。这两个角度在空间中关系如图 9.14 所示。

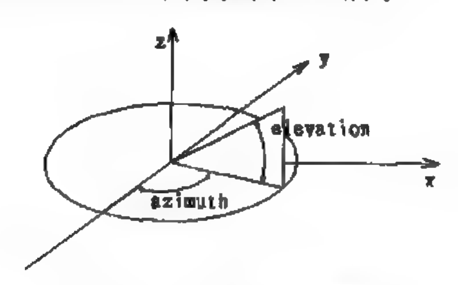


图 9.14　az，el 空间关系图

第 3 种格式的作用是在直角坐标系内设置观察点的位置(x，y，z)。

第 4 种格式的作用是产生俯视二维图，此时 az = 0，el = 90。

第 5 种格式的作用是将观察点设在系统值值处：az = -37.5，el = 30。

第 6 种格式的作用是通过一个 4×4 的透视变换矩阵设置观察角度。

第 7，8 种格式的作用分别是查询当前视图的视角和透视变换矩阵。

例 14

```
subplot(2, 2, 1)
mesh(peaks)
[az1, el1]=view
subplot(2, 2, 2)
mesh(peaks)
view(20, 45)
subplot(2, 2, 3)
```

```
mesh(peaks)
view([50, 50, -10])
subplot(2, 2, 4)
mesh(peaks)
view(2)
```

程序运行结果为：

```
az1 =
      -37.5
el1 =
       30
```

绘得结果如图 9.15 所示。

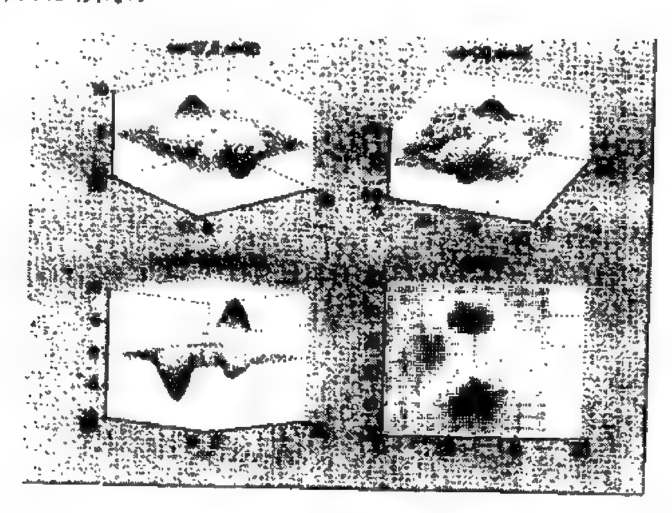


图 9.15　视角变换图

2. 三维图形的透视 hidden

在 MatLab 中，用 mesh 命令绘制网格图时，系统在默认情况下会消隐掉重叠在后面的网格，利用透视命令 hidden 可以看到被掩盖的部分。

hidden 命令的使用很简单，它只是掩盖开关命令。其使用格式为：

```
hidden on
hidden off
```

命令 hidden off 的作用就是关闭消隐命令，从而能看到被挡住的部分；命令 hidden on 是打开消隐命令，使图中前面的部分挡住后面的部分。

例 15

```
mesh(peaks)
hidden off
```

绘得结果如图 9.16 所示。

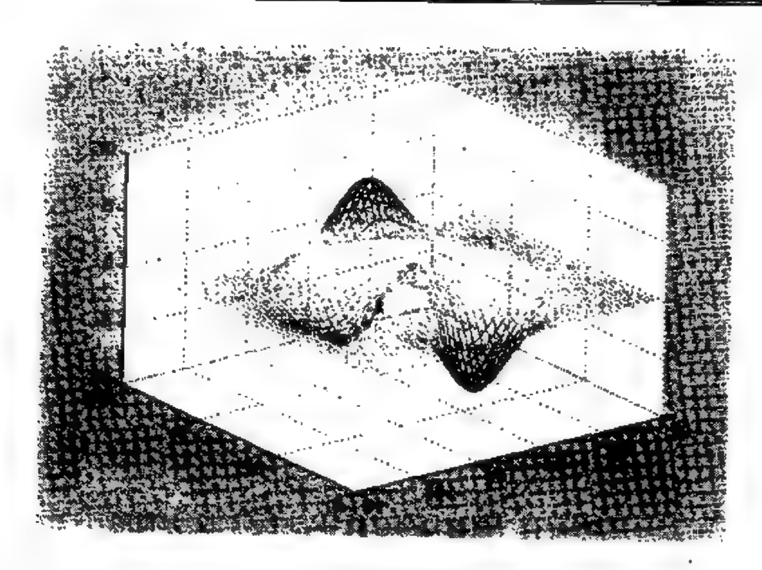


图 9.16　网格消隐图

3. 曲面图形颜色的遮掩 shading

shading 命令用于曲面图形颜色不匀时的均衡处理。它的使用格式为：

```
shading flat
shading faceted
shading interp
```

其中第 1 种格式将根据整个网格的值在其每个网眼上确定一个标志颜色的值，由于相邻网眼的值是相近的，因此其颜色也较相近。

第 2 种格式是 shading 命令的默认格式，它对网眼的颜色不作处理，但将加深网线的黑色。

第 3 种格式将在网眼内采用内差法详细计算网眼内不同位置处的颜色差异，由此法绘得的图形的颜色最连贯。

例 16

```
surf(peaks(80))
shading flat
figure
surf(peaks(80))
shading faceted
figure
surf(peaks(80))
shading interp
```

绘得结果如图 9.17，9.18，9.19 所示。

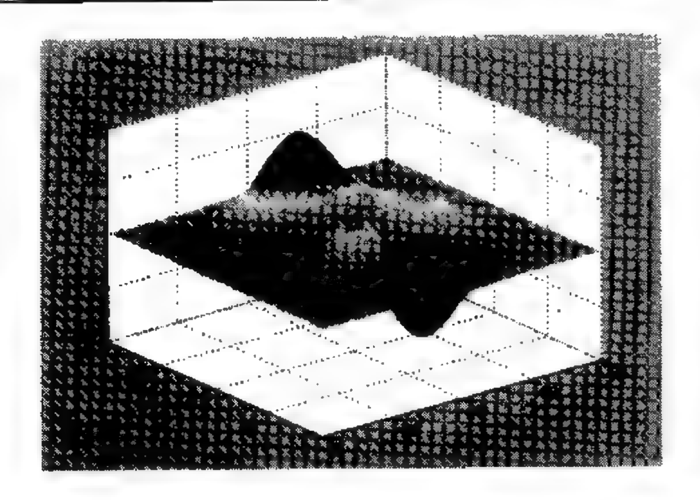


图 9.17　函数 peaks 的 shading flat 效果图

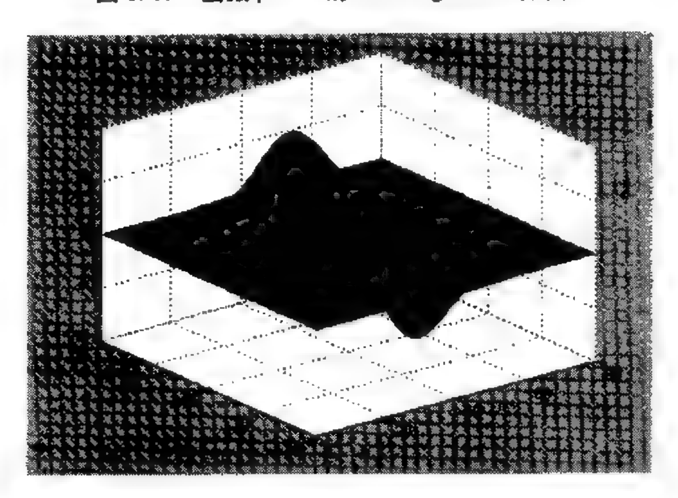


图 9.18　函数 peaks 的 shading faceted 效果图

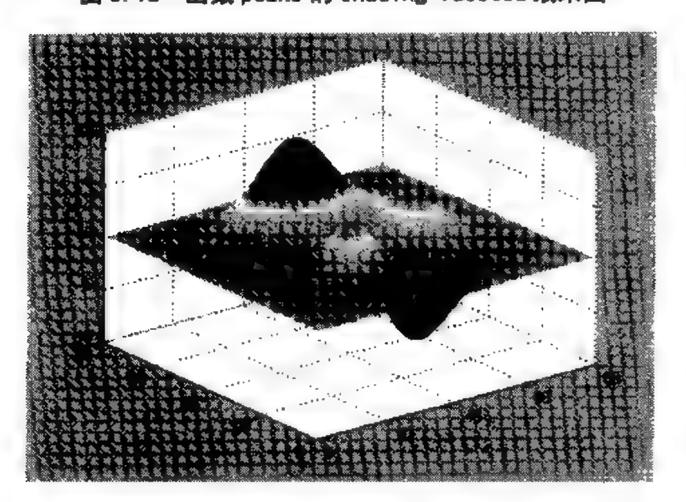


图 9.19　函数 peaks 的 shading interp 效果图

9.3.4　三维图形的光照控制

与 view 命令相似，MatLab 中还有一个用来控制光源位置和光线性质的命令 surfl。这个命令用来产生带有明暗效果的三维表面图，其使用格式为：

```
surfl(z)
surfl(x, y, z)
surfl(x, y, z, s)
surfl(x, y, z, s, k)
```

其中参数 x，y，z 是用来绘制曲面的数值矩阵，对其要求与在 surf(x，y，z)中的要求完全相同。

参数 s 用来设定光源的位置，其默认值为光源的方位角 az 在观察点逆时针方向的 π/4 处。在设定光源位置时，既可以采用直角坐标系，也可以采用球坐标系。在直角系中，s=[sx，sy，sz]，即 s 为光源的坐标位置，在球坐标系中，s=[az，el]，即光线的投射角度。

参数 k 是控制光线性质的向量，k=[ka，kd，ks，spread]。k 用来给定三维图形的背景光(ambient)，漫射光(diffuse)，定向光(specular)在总投射光中各占比例。元素 spread 表示的是图像的扩散系数。由于 k 的设定较复杂，因此在一般情况下默认，其默认值为[0.55，0.6，0.4，10]。

例 17

```
x=-3: 0.1: 3;
[x, y]=meshgrid(x);
surfl(x, y, peaks(x, y), [45, 45])
figure
x=-3: 0.1: 3;
[x, y]=meshgrid(x);
surfl(x, y, peaks(x, y), [45, 45], [0.1, 0.1, 0.1, 1])
```

绘得结果如图 9.20，9.21 所示。

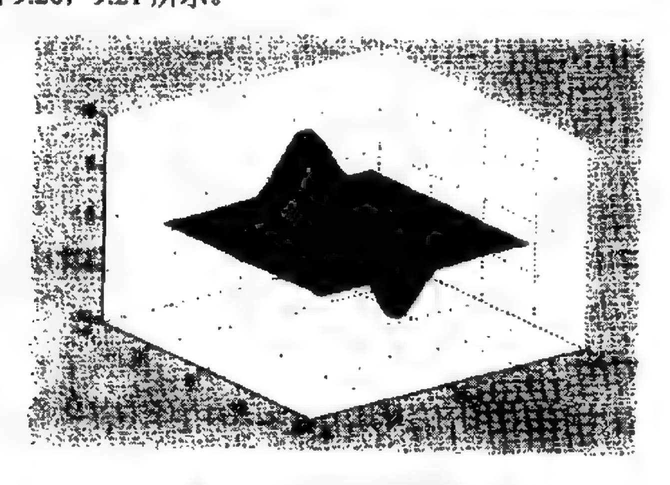


图 9.20　三维光照控制图 1

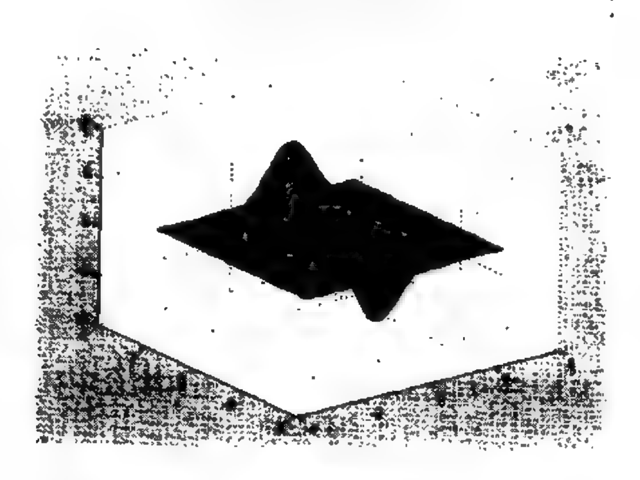


图 9.21 三维光照控制图 2

9.3.5 柱面和球面的表达

cylinder 和 sphere 命令是 MatLab 提供的两个用来绘制柱面和球面的命令。只需 1、2 行简单的命令，就可完成一幅漂亮的彩图。

- 柱面的表达

与平时作图一样，绘制一个柱面首先要给出它的母线和轴线。在 cylinder 命令中，柱面的轴线已经定为 z 轴，因此只要给出其母线的描述就可完成一个柱面，再加上一个参数 n，用来描写旋转柱面上的分格线条数，就可完成一个完美的柱面。

其使用格式为：

```
[X, Y, Z]=cylinder(r, n)
```

其中参数 r 为一向量，用来描述柱面母线。

[X，Y，Z] 是该命令的运行结果，应用 mesh(X，Y，Z)命令可将所画图形重画出来。

例 18

```
t=pi*2: pi/12: 5*pi;
r=sin(t)+t;
cylinder(r, 100)
```

结果如图 9.22 所示。

- 球面的表达

由于绘制的只是一个单位球面(即其半径为 1)，因此绘制球面比绘制柱面简单，只需设置分格线条数 n 即可。其使用格式与柱面命令 cylinder 完全相同，如下所示：

```
[X, Y, Z]=sphere(n)
```

但因在使用中，很少单独使用一个球面，因此该命令常和其他命令一起完成对一组数据的表达。

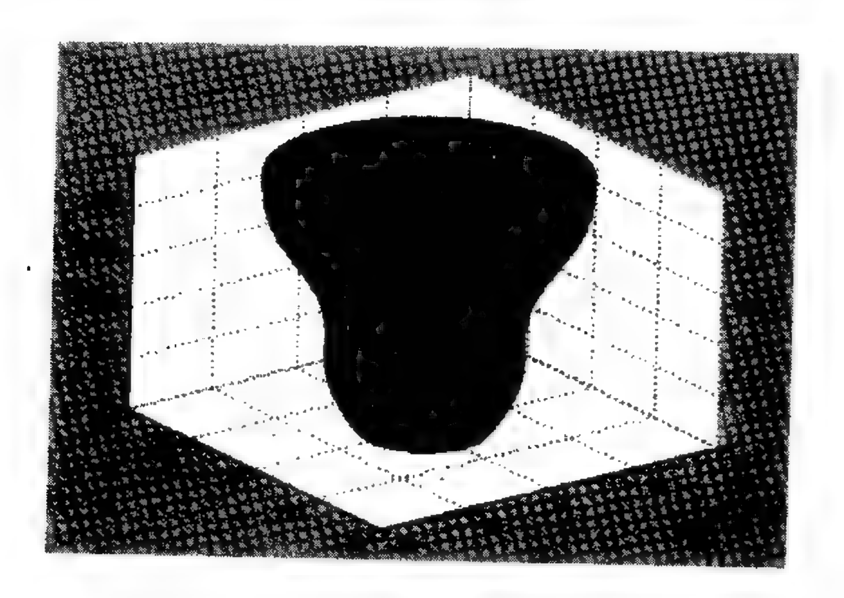


图 9.22　柱面图

例 19

```
[X, Y, Z]=sphere(50);
Tem=[0: 1: 25, 25: -1: 1];
T=meshgrid(Tem);
surf(X, Y, Z, a)
```

此处命令涉及到色图的应用，将在本章第 9 节详细介绍

结果如图 9.23 所示。

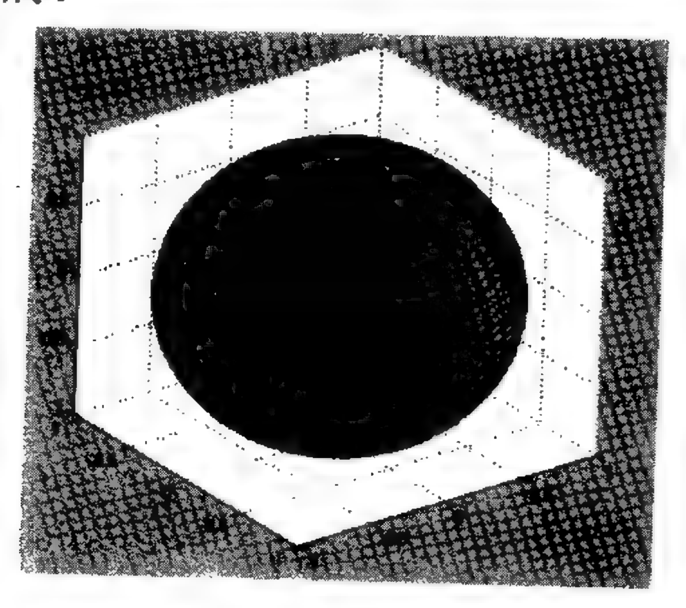


图 9.23　球面图

9.4 特 殊 图 形

在工程计算中，有时为了将抽象的数据表达得更形象，除了绘制出它的二维、三维以至四维图形外，还经常用到诸如直方图、面积图、饼图、阶梯图、火柴图、误差棒棒图、拓扑图、统计频数直方图等特殊图形。本节将就介绍这些图形的绘制命令。

● 直方图 bar

在直方图里，还可如下细分出 4 个小类，并相应各有一个专用命令，如表 9.4 所示。

表 9.4 直方图命令在不同情况下的详细名称

图形名称	二维直方图	三维直方图
竖直直方图	bar	bar3
水平直方图	barh	bar3h

这 4 个命令虽然名称有所不同，但其使用格式是完全相同的，现以 bar 命令为例，介绍它们的使用方法。

bar 命令的使用格式有如下几种：

```
bar(Y)
bar(x, Y)
bar(..., width)
bar(..., 'style')
bar(..., LineSpec)
```

前面两种格式是最常用的，当只有参数 Y 时，系统将以 Y 的下标作为直方图的横坐标值；当有参数 x 时，将以向量 x 的值为横坐标值，不过此时要求向量 x 必须是严格单调增或单调减的。

第 3 种格式将确定直方图每个数据条在图上的宽度，其默认值为 0.8，但当其值大于 1 时，2 组不同的数据条将重合在一起。

第 4 种格式用于确定直方图的排列形式，有分离式(detached)、组合式(grouped)及堆栈式(stacked)，其中分离式为默认值。

第 5 种格式中的 LineSpec 是用于控制直方图颜色的参数。

它们的具体使用如例 20 所示。

例 20

```
subplot(3, 2, 3)
bar(y, 'grouped')
subplot(3, 2, 4)
bar(y, 'stacked')
subplot(3, 2, 5)
bar(y, 0.3)
subplot(3, 2, 6)
bar(y, 1.3)
```

结果如图 9.24 所示。

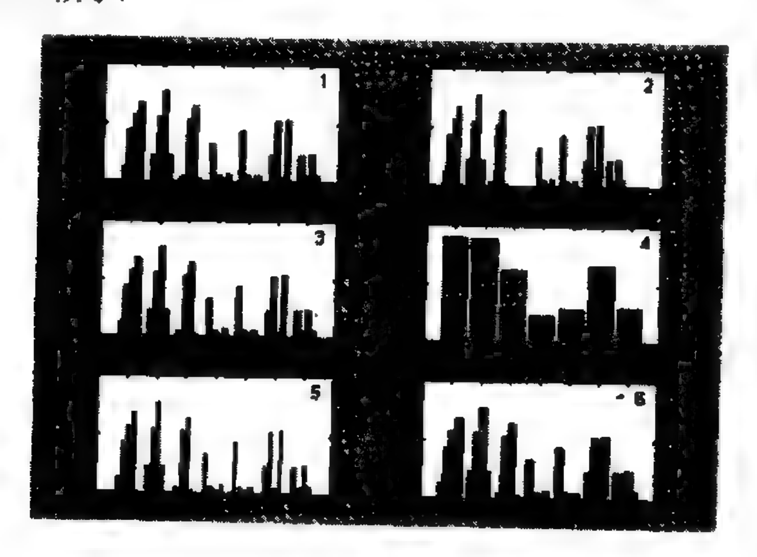


图 9.24　直方图

● 面积图 area

在实际应用中，面积图适用于表现各个不同部分对整体所作的贡献，因此使用范围十分广泛。下面是绘制面积图的命令 area 的使用方法和格式：

```
area(Y)
area(x, Y)
area(..., leval)
```

前面两种使用格式和 plot(x，Y)命令的使用完全一样，只不过在绘制得的图形中，area 命令将连线图到 x 轴的那部分填上了颜色。

第 3 种用法唯一与前面 2 个命令不同的是，填色部分改为由连线图到 y=leval 的水平线之间的部分。

例 21

```
x=0: 0.01: 2;
Y=[x; x.^0.2; x.^0.6]';
subplot(2, 1, 1)
area(Y)
subplot(2, 1, 2)
area(x, Y, 2)
```

结果如图 9.25 所示。

● 饼图 pie

一个向量或矩阵各元素在所有元素之和中所占的比例，可用饼图来表示。同直方图命令 bar 相似，饼图命令 pie 也有两个分别适用于二维和三维情况的专用命令：pie 和 pie3。

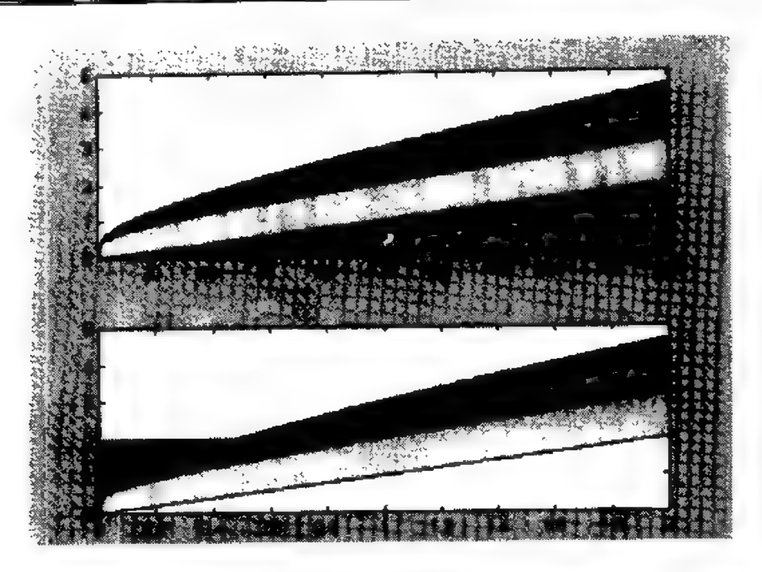


图 9.25 面积图

由于 2 个命令的使用方法完全一样，下面以 pie 命令为例，介绍其使用格式：

```
pie(X)
pie(X, explode)
```

其中 X 为饼图所要表现的向量或元素，命令 pie 将把 X 的每一个元素在所有元素总和中占的比例表达出来。参数 explode 为一与 X 同维的矩阵，当所有元素都为零时，饼图的各个部分将连在一起组成一个圆，而当其中有非零元素存在时，X 阵中的相应位置的元素在饼图中对应的扇形将向外移出一些，加以突出。

例 22

```
X=[23 43 13;
   22 40 29];
explode=[0 0 1;
   1 0 0];
pie(X, explode)
```

结果如图 9.26 所示。

另外，当矩阵 X 的所有元素之和不足 1 时，饼图命令 pie 将以各元素值作为其在饼图中所占比例绘制，所得结果是一个不完整的饼图。

例 23

```
X=[0.1 0.12 C.21 0.34];
pie3(X)
```

结果如图 9.27 所示。

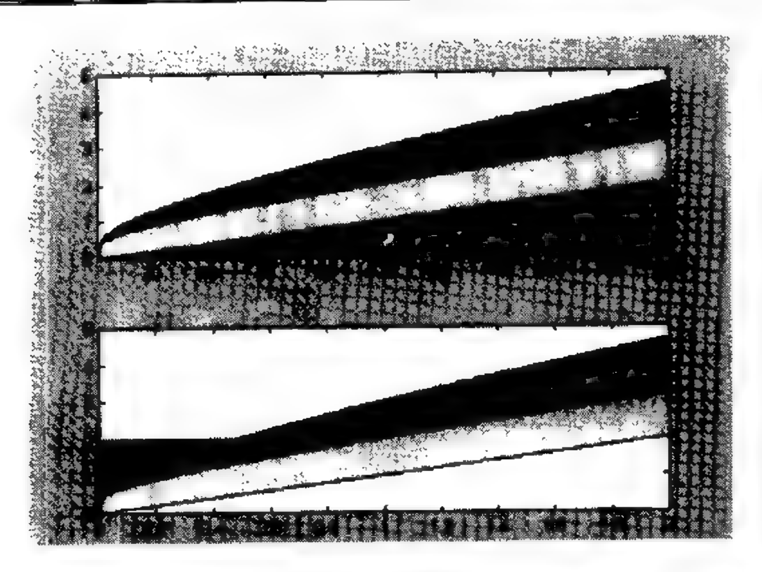


图 9.25　面积图

由于 2 个命令的使用方法完全一样，下面以 pie 命令为例，介绍其使用格式：

```
pie(X)
pie(X, explode)
```

其中 X 为饼图所要表现的向量或元素，命令 pie 将把 X 的每一个元素在所有元素总和中占的比例表达出来。参数 explode 为一与 X 同维的矩阵，当所有元素都为零时，饼图的各个部分将连在一起组成一个圆，而当其中有非零元素存在时，X 阵中的相应位置的元素在饼图中对应的扇形将向外移出一些，加以突出。

例 22

```
X=[23 43 13;
   22 40 29];
explode=[0 0 1;
   1 0 0];
pie(X, explode)
```

结果如图 9.26 所示。

另外，当矩阵 X 的所有元素之和不足 1 时，饼图命令 pie 将以各元素值作为其在饼图中所占比例绘制，所得结果是一个不完整的饼图。

例 23

```
X=[0.1 0.12 C.21 0.34];
pie3(X)
```

结果如图 9.27 所示。

例 24

```
x=-3: 0.1: 3:
y=randn(1000, 1);
hist(y, x)
figure(2)
theta=y*2*pi;
rose(theta)
[n, xout]=hist(y, 10)
```

结果为:

```
n =
    5    10    44   116   163   269   226   106    48    13
xout =
        -3.16857928880728
        -2.53767242167393
        -1.90676555454058
        -1.27585868740723
        -0.644951820273886
        -0.0140449531405379
         0.616861913992811
         1.24776878112616
         1.87867564825951
         2.50958251539286
```

所绘图如图 9.28 和 9.29 所示。

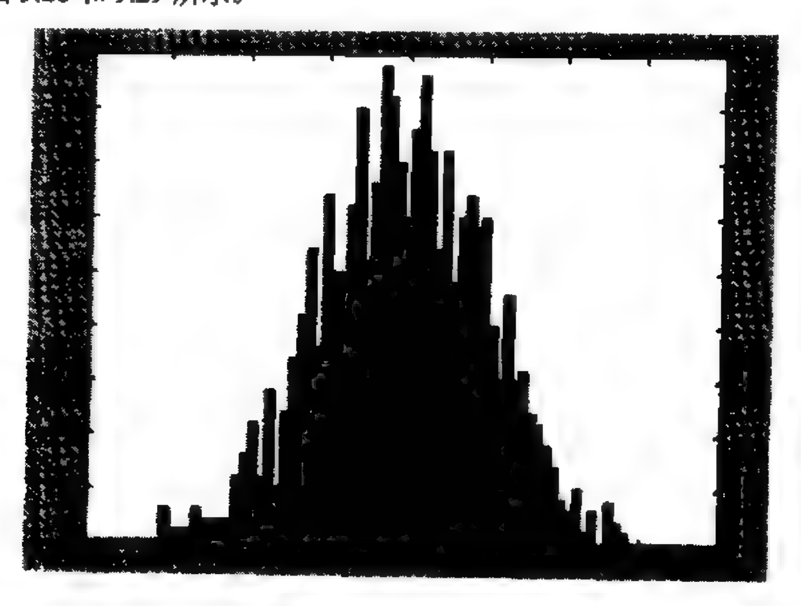


图 9.28 柱状图 1

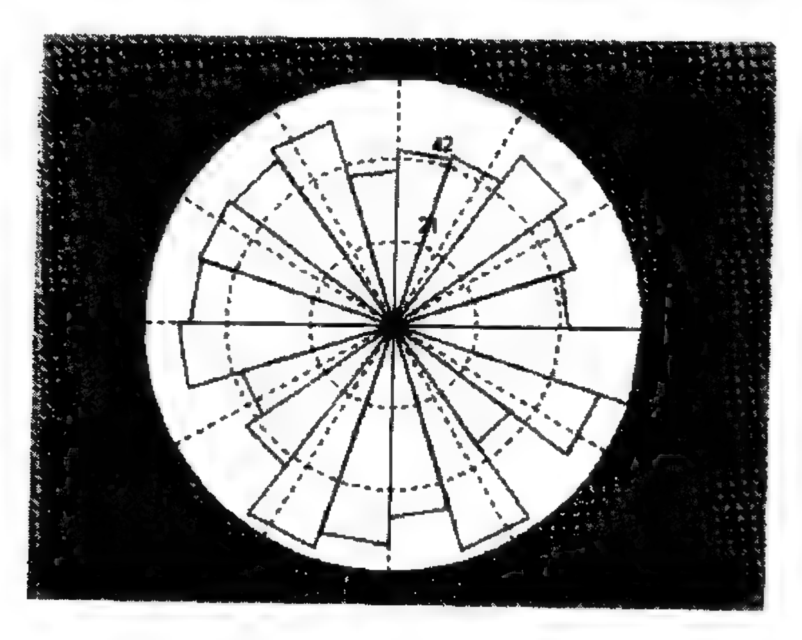


图 9.29　柱状图 2

- 拓扑图 gplot

绘制拓扑图的命令是 gplot，其使用格式为：

```
gplot(A, C)
gplot(A, C, 'color_linestyle_maker')
```

其中 A 为一 n×n 邻阶矩阵，C 为一个 n×2 或 n×3 的相配矩阵，其中 n 为拓扑图中节点个数，C 是 n×2 或 n×3 的相配矩阵决定了每个节点与其他 2 个或 3 个节点相连。'color_linestyle_maker'则起与在 plot 命令中相同的控制图形表现的作用。

例 25

```
k = 1: 50;
[B, XY] = bucky;
gplot(B(k, k), XY(k, : ), 'r-p')
```

结果如图 9.30 所示。

- 火柴杆图 stem

火柴杆图是因所绘得图形的形状酷似一根根火柴而得名，这种绘图方法适用于非连续数据。它在二维平面中的使用格式如下：

```
stem(Y)
stem(X, Y)
stem(..., 'fill')
stem(..., LineSpec)
```

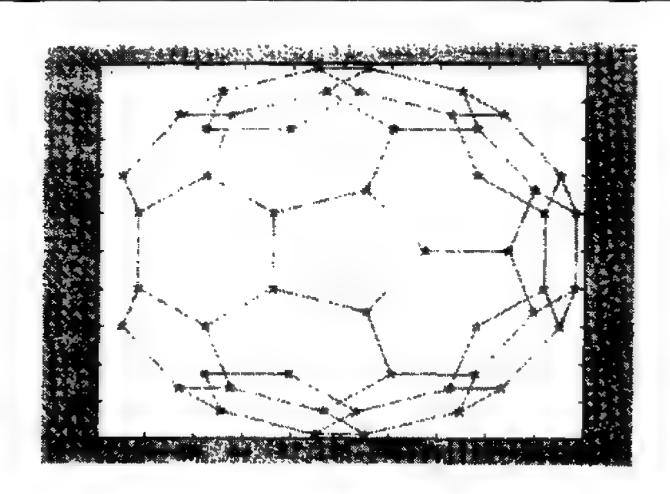


图 9.30 拓扑图

前两种格式和最后 1 种格式对参数 X，Y 的要求与 plot 命令相同(其中的 LineSpec 就是'color_linestyle_maker')，不过所绘出的图形不是连线图。第 3 种格式中参数'fill'的作用是将“火柴头”填上颜色。

例 26

```
x=0: 0.2: 4;
y=sin(x).*exp(cos(x));
stem(x, y, 'fill', 'r-p')
```

结果如图 9.31 所示。

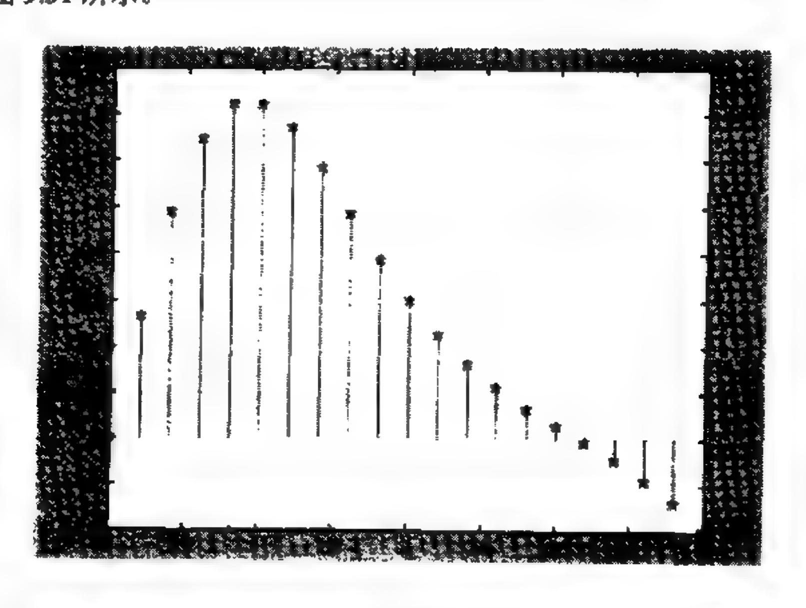


图 9.31 火柴杆图

火柴杆图在三维空间中还有一个专用命令：stem3。它的使用格式为：

```
stem3(Z)
stem3(X, Y, Z)
```

```
stem3(..., 'fill')
stem3(..., LineSpec)
```

式中对参数X，Y，Z的要求与在plot3，mesh，surf命令中的要求完全相同。参数'fill'，LineSpec的含义同stem命令。

例 27

```
x=0: 0.1: 5;
y=0: 0.1: 5;
z=x.^2+y.^2;
stem3(cos(x), sin(y), z, '-bd')
view(160, 30)
```

结果如图9.32所示。

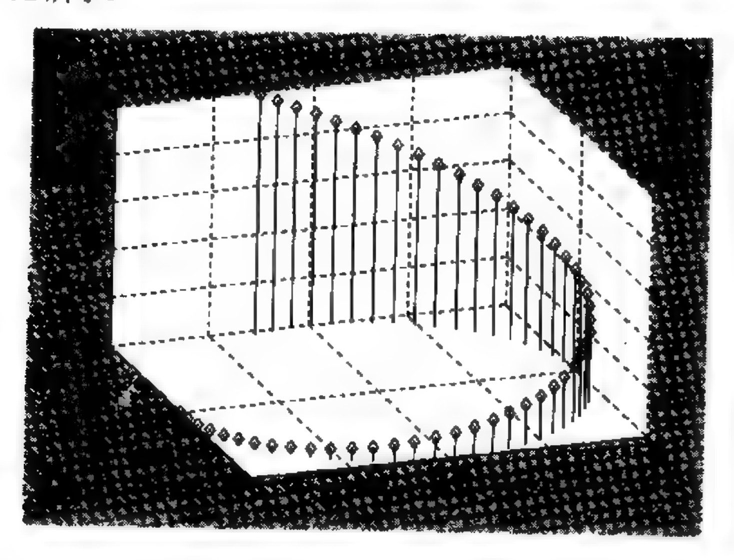


图 9.32 三维火柴杆图

- 阶梯图 stairs

阶梯图与火柴杆图类似，均是表现间断数据的工具。绘制阶梯图的命令stairs的使用与stem命令基本相同，其使用格式为：

```
stairs(Y)
stairs(X, Y)
stairs(..., LineSpec)
[xb, yb] = stairs(Y)
[xb, yb] = stairs(X, Y)
```

前面3种格式与stem命令中的相同格式对参数的要求完全一样。后面2个命令格式并不画出阶梯图，而是将阶梯图上各点的坐标值赋予变量xb和yb，可用plot(xb，yb)命令制得阶梯图。

例 28

```
x=0: 0.2: 2*pi;
y=sin(x).*x;
```

```
stairs(x, y)
```

绘得结果如图 9.33 所示。

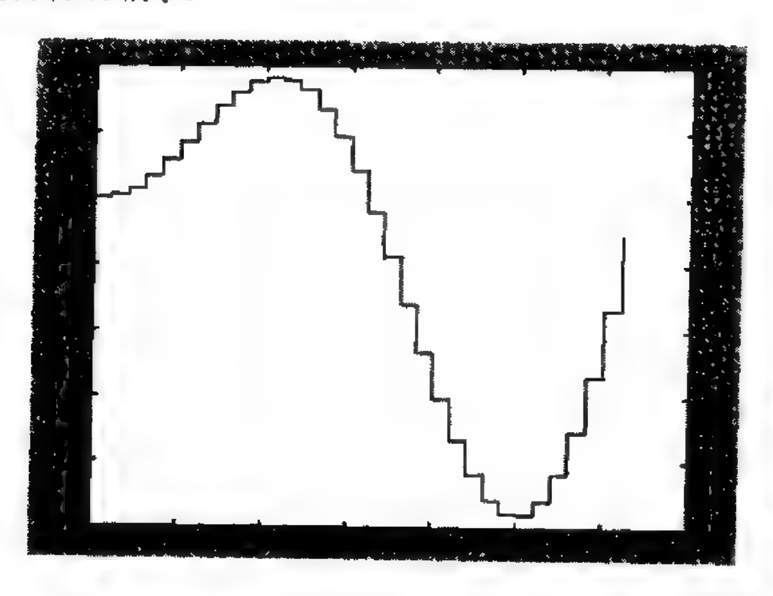


图 9.33 阶梯图

● 复数(向量)的平面表达

在 MatLab 中，有 2 个用来在平面中表达复数(向量)的命令：compass 和 feather，利用这 2 个命令可以很方便的表示出复数(向量)的方向特性。它们的使用格式为：

```
compass
compass(X, Y)
compass(Z)
compass(..., LineSpec)
feather
feather(X, Y)
feather(Z)
feather(..., LineSpec)
```

它们的作用虽然都是描述复数(向量)，但采取的方法不同。compass 命令以原点为起点绘制每个复数(向量)，feather 命令则以点(1, 0)作为第 1 个复数(向量)的起点，点(2, 0)作为第 2 个复数(向量)的起点，…，点(n, 0)作为第 n 个复数(向量)的起点。

在这 2 个命令的参数中，X，Y 均为实数矩阵。其中 X 是复数的实部，Y 是复数的虚部。Z 为虚数矩阵。LineSpec 是对图形基本参数的控制字符。

例 29

```
x=0: pi/15: 2*pi;
compass(sin(x), x.*cos(x)/5)
figure
feather(sin(x), cos(x))
```

结果如图 9.34，9.35 所示。

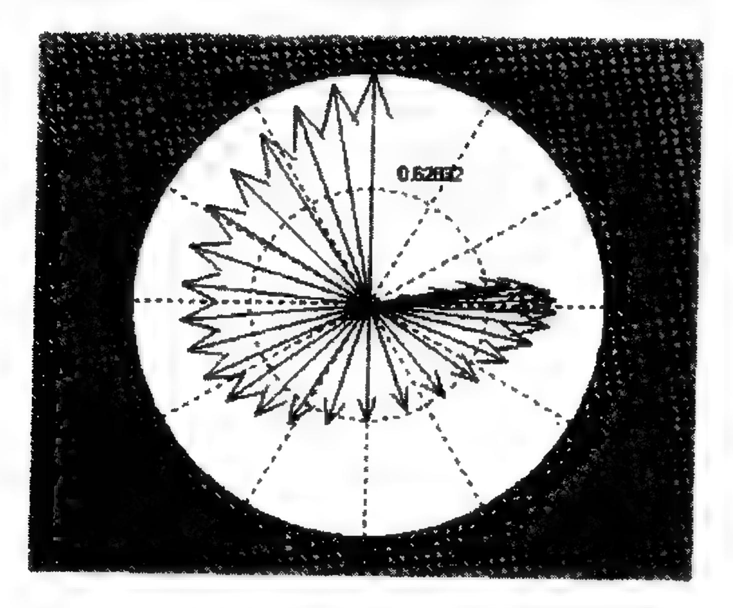


图 9.34　指针图

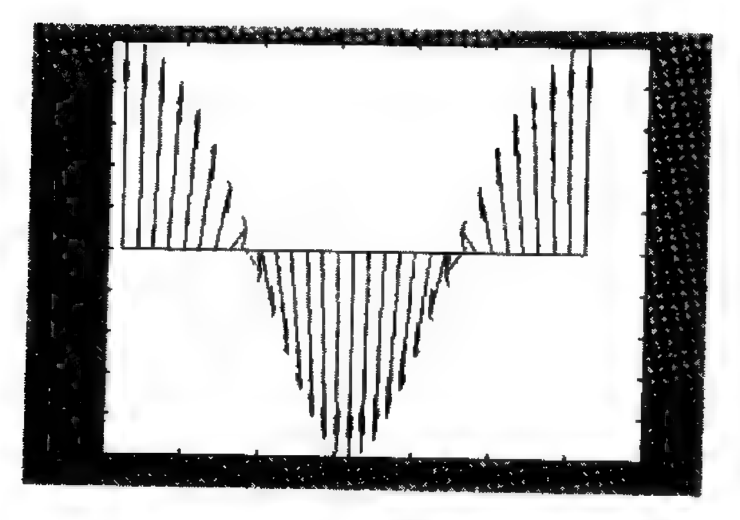


图 9.35　羽毛图

- 二维、三维向量场的表现 quiver

仅仅能够在图中表现有限几个向量是远远不够的，在实际应用中经常会遇到如大空间磁场分布等向量场，这时如果能在图中形象的画出其向量场的形状，那将对解决问题有很大的帮助。quiver 和 quiver3 命令正是适用于二维和三维情况的现代两个命令。

quiver 命令的使用格式为：

```
quiver(U, V)
quiver(X, Y, U, V)
quiver(..., scale)
quiver(..., LineSpec)
quiver(..., LineSpec, 'filled')
```

如果 U，V 均为 m×n 矩阵，则命令 quiver(U，V)将在 X= ：n，Y=1：m 的平面内绘制出在每一点(x，y)处由 U，V 决定的向量。当参数中给出了向量 X，Y 时，命令 quiver(X，Y，U，V)将在由 X，Y 决定的每一个平面点上画出相应向量。

参数 scale 是用来控制看到的图中向量“长度”的实数。有时在图中由于每点向量"过长"导致向量彼此重叠掩盖，这时就需将 scale 由默认的 1 改小；相反，就把它变大。

LineSpec，'filled' 2 个参数用来控制图形的表现。

例 30 绘制位于原点的半径为 1 的圆柱的近似引力场

```
[x, y]=meshgrid(-0.5: 0.05: 0.5);
for i=1: 21
  for j=1: 21
    a=x(i, j).^2+y(i, j).^2;
    if a<0.1
      x(i, j)=nan;
      y(i, j)=nan;
    end
    z(i, j)=1./(x(i, j).^2+y(i, j).^2).^1.5;
    if z(i, j)>100
      z(i, j)=nan;
    end
  end
end
[px, py]=gradient(z, .05, .05);
contour(x, y, z)
hold on
quiver(x, y, px, py)
```

结果如图 9.36 所示。

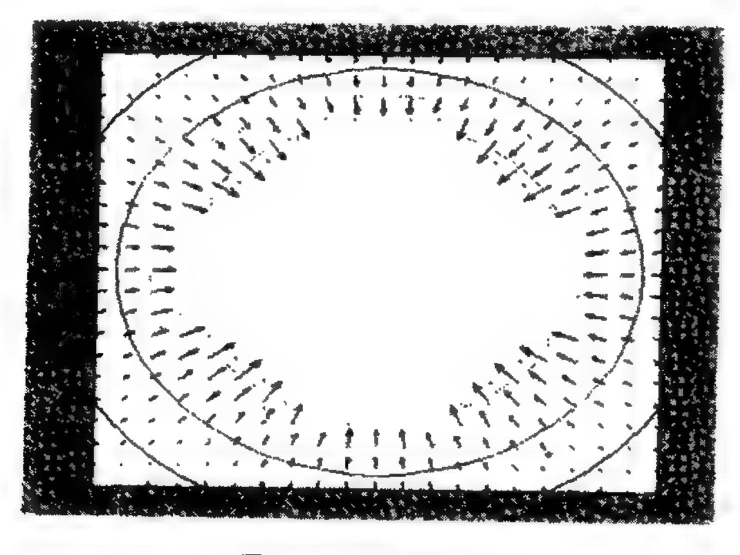


图 9.36 引力场示意图

请注意程序中用非数 nan 裁剪图形的方法。

quiver3 命令的使用格式为：

```
quiver3(Z, U, V, W)
quiver3(X, Y, Z, U, V, W)
quiver3(..., scale)
quiver3(..., LineSpec)
quiver3(..., LineSpec, 'filled')
```

上面前两种格式的作用是分别在 mesh(Z)和 mesh(X, Y, Z)的节点上绘制由矩阵 U, V, W 决定的向量．但一般可由 gradient 和 surfnorm 命令计算相应曲线或曲面上的切线及法线方向。后面 3 个使用格式加了与前面相同的参数。

例 31 试画出下面计算得的曲面上的法线方向向量

```
theta=0: pi/10: 2*pi;
rho=0: 0.5: 5;
[t, r]=meshgrid(theta, rho);
z=r.^2;
[x, y, z]=pol2cart(t, r, z);
surf(x, y, z)
hold on
[U, V, W]=surfnorm(x, y, z);
quiver3(x, y, z, U, V, W);
```

结果如图 9.37 所示。

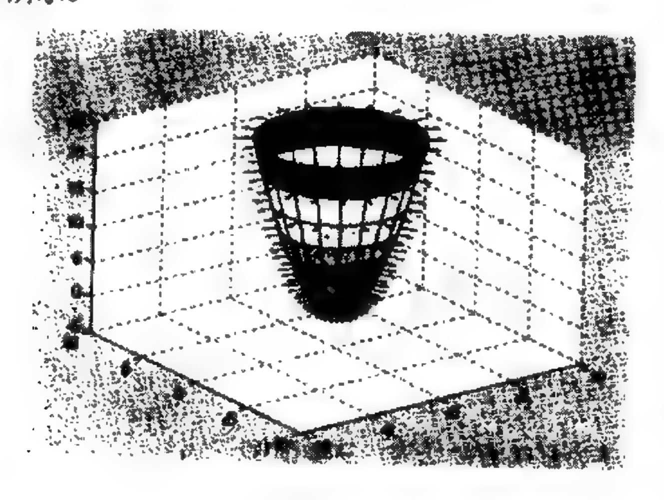


图 9.37 空间向量图

- 误差棒图 errorbar

误差棒图适用于表现数据的可信度或关于一个已知标准的偏离程度。命令 errorbar 的使用格式为：

```
errorbar(Y, E)
errorbar(X, Y, E)
errorbar(X, Y, L, U)
```

```
errorbar(..., LineSpec)
```

式中参数 X，Y，E，U，L 均必须为 n×m 矩阵或 n 维向量。

第 1、2 种格式的作用是在 plot(Y)，plot(X，Y)的基础上，在图中每个 Y 元素所对应的点上画出一根误差棒。误差棒的长度为 2E(i)，即误差棒的中点就在(X，Y)上。

第 3 种格式更详细的设置了误差棒的上下 2 部分的长度，L 是误差棒下端到(X，Y)的距离，U 是误差棒上端到(X，Y)的距离。

例 32

```
x=[0: 0.2: 4*pi; 1: 0.2: 4*pi+1];
y=[sin(x)];
e=[0: 1/(length(x)-1): 1; 0: 1/(length(x)-1): 1];
errorbar(x, y, e)
```

绘得结果如图 9.38 所示。

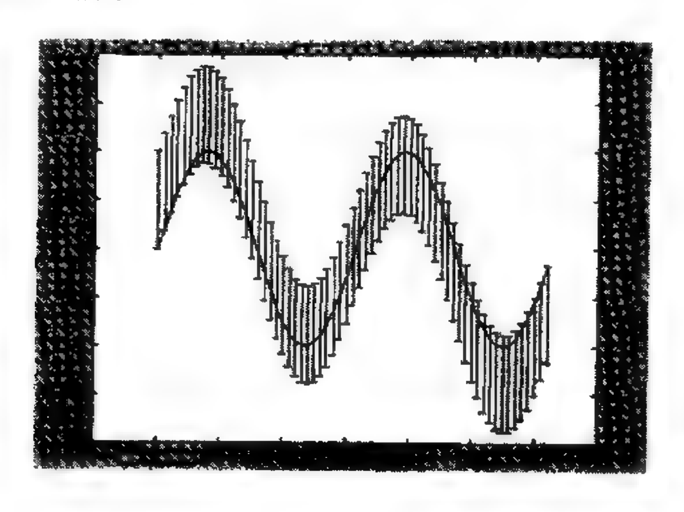


图 9.38 误差图

9.5 二元函数、三元函数的图像表示

9.5.1 在直角坐标系下绘制函数的二维、三维图形

1. meshgrid 命令

由二元函数、三元函数的表达式 z=f(x，y)和 w=f(x，y，z)可知，需要在三维空间表达二元函数、在四维空间表达三元函数。由对应关系 z(i，j)=f(x(i)，y(j))和 w(i，j，k) =f(x(i)，y(j)，z(k))需要用 for 循环语句才能完成对每个节点的计算。meshgrid 命令可以将表示在坐标轴上点位置的向量 x，y 及 z 转化为空间节点坐标(x(i，j)，y(i，j))和(x(i，j，k)，y(i，j，k)，z(i，j，k))，有了节点坐标，在计算中就可以充分发挥 MatLab 的矩阵计算能力，利用

下面的表达式：

```
z(i, j)=f(x(i, j), y(i, j));
 w(i, j, k)= (x(i, j, k), y(i, j, k), z(i, j, k))
```

从而可像计算一元函数一样完成对多元函数的计算，而不再利用 for 循环。

Meshgrid 命令的使用格式有两种。

(1) 在绘制二元函数时，只需产生的节点都在一个平面上，命令为：

```
[X, Y]=mesh(x, y)
```

如果 x 为 m 维向量，y 为 n 维向量，则产生的 X，Y 为 n×m 矩阵。

例 33

```
x=[1 2 3 4];
y=[11 12 13 14 15 16];
[X, Y]=meshgrid(x, y)
```

结果为：

```
X =
          0.1     0.5     0.9     1.3
          0.1     0.5     0.9     1.3
          0.1     0.5     0.9     1.3
          0.1     0.5     0.9     1.3
          0.1     0.5     0.9     1.3
          0.1     0.5     0.9     1.3
Y=
          1.1     1.1     1.1     1.1
          1.5     1.5     1.5     1.5
          1.9     1.9     1.9     1.9
          2.3     2.3     2.3     2.3
          2.7     2.7     2.7     2.7
          3.1     3.1     3.1     3 .1
```

对这个结果可用 plot 命令看看其节点的分布：

```
plot(X, Y)
```

结果如图 9.39 所示。

(2) 绘制三元函数时，产生的节点将分布在一个空间内，其使用格式为：

```
[X, Y, Z]=meshgrid(x, y, z)
```

例 34

```
x=0.3: 0.4: 2.5;
y=0.1: 0.2: 2;
z=0.4: 0.1: 0.7;
[X, Y, Z]=meshgrid(x, y, z);
figure
hold on
```

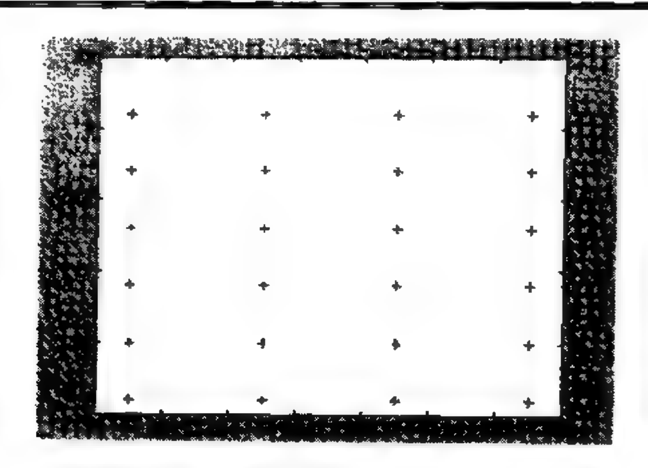


图 9.39 命令 meshgrid 在平面上产生的点

```
n=0;
for i=1: length(x)
   for j=1: length(y)
      for k=1: length(z)
         if k==1
            str='r+'
         elseif k==2
            str='go'
         elseif k==3
            str='bp'
         elseif k==4
            str='y.';
         end
         plot3(X(j, i, k), Y(j, i, k), Z(j, i, k), str)
      end
   end
end
```

结果如图 9.40 所示。

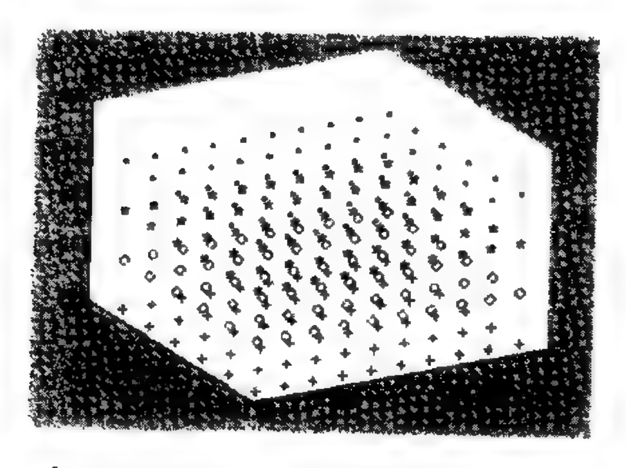


图 9.40 命令 meshgrid 在三维空间内产生的点

可见，命令 meshgrid(x，y，z)产生了作图所需的每一个节点。

2. **绘制二元函数 z=f(x，y)的函数图**

有了 meshgrid 命令和 mesh，surf 命令，绘制二元函数图就非常简单。

例 35

```
x=-3: 0.05: 3;
y=-3: 0.05: 3;
[X, Y]=meshgrid(x, y);
z=0.1*sin(X.^2+Y.^2);
mesh(x, y, z)
```

结果如图 9.41 所示。

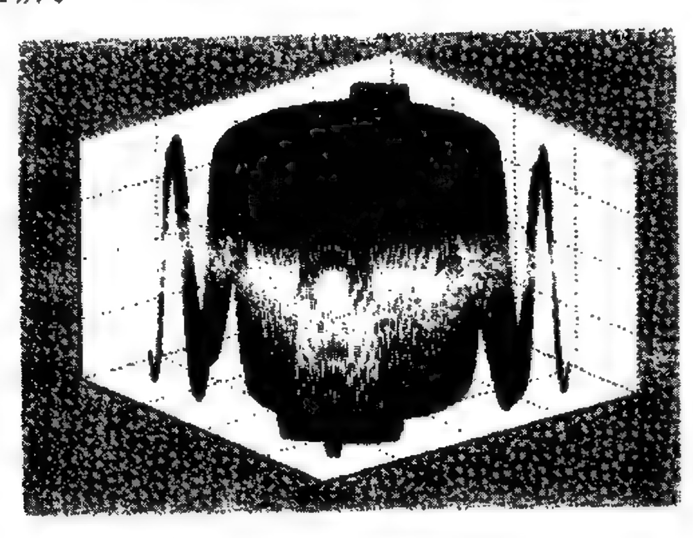


图 9.41 三元函数 $z=0.1\sin(X^2+Y^2)$ 的图像

附：三维曲面等高线的绘制

在 MatLab 中，可以利用 contour，contour3 命令，任意在平面或空间中表现出二元函数的图像——三维曲面的等高线。

绘制平面等高线 contour 命令的具体使用格式为：

```
contour(Z, n)
contour(x, y, Z, n)
contour(x, y, Z, v)
contourf(...)
```

Z 为二元函数的函数值矩阵。x，y 为其横纵坐标值向量。

参数 n 为整数，指定了绘出等高线的条数。

参数 v 为向量，指定了在哪些高度绘出等高线，如只想在一个高度 z 绘出等高线，则 v=[z z]。

contourf(...)命令的参数与 contour 命令完全相同，只是其绘出的等高线图将被自动填上颜色。

C=contour(x，y，Z，n)

C=contour(x，y，Z，v)

上面 2 个命令用来计算所画等高线的 x，y 坐标值。

```
clabel(C)
clabel(C, v)
clabel(C, 'manual')
```

上面 3 个命令用来标注计算的 C 阵处的高度值。clable(C)将把所绘等高线全部自动标注，clable(C，v)将自动标注由向量 v 确定的若干条等高线的高度值，此处的向量 v 必须是前面 contour 命令中 v 的子集。命令 clable(C，'manual')是手工标注高度。

例 36

```
Z=peaks;
c=contour(Z, 5);
clabel(c, 'manual')'
```

结果如图 9.42 所示。

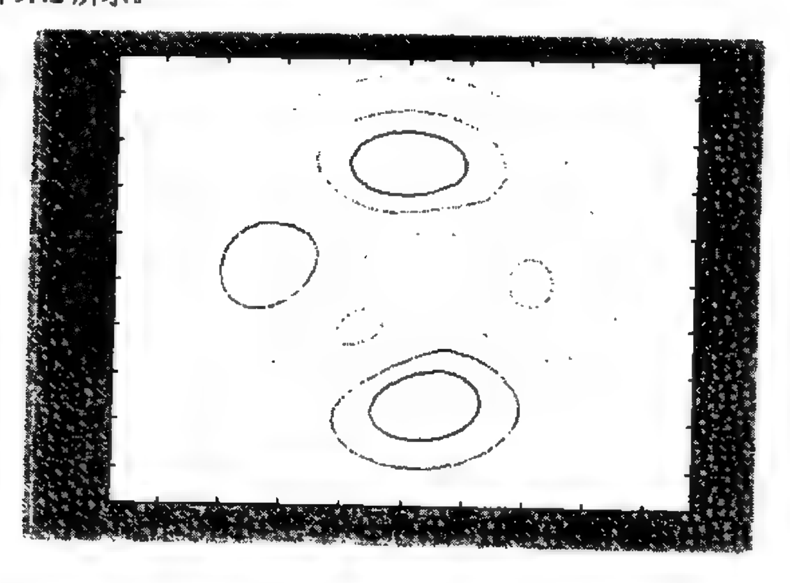


图 9.42 peaks 的等高线

绘制空间等高线 contour3 的使用格式与 contour 命令完全类似，这里不再赘述。具体应用见下例。

例 37

```
Z=peaks;
c=contour3(Z, 15);
clabel(c, 'manual')'
```

结果如图 9.43 所示。

3. 绘制三元函数 w=f(x，y，z)的函数图

由于三元函数的定义域是整个三维空间，因此再没有什么空间量来表示函数的值了，只有用颜色来表示这存在于第四维空间的值。在此，除了运用 meshgrid 命令来创建空间节点，还要用到一个新命令 slice：四维表现和切片图。

Slice 命令的使用格式为：

```
slice(X, Y, Z, W, xi, yi, zi)
```

式中各个参数含义为：

X，Y，Z　　由 meshgrid(x，y，z)确定的空间节点矩阵。

W　　在节点上的三元函数的函数值。

xi，yi，zi　　确定分别垂直于 x，y，z 轴的切面到原点的垂直距离，可为向量。

例 38

```
x=-2: 0.1; 2;
y=0: 0.1: 4;
z=-2: 0.1: 2;
[X, Y, Z]=meshgrid(x, y, z);
W=X.*sin(-X.^2.*Y-Y.^2+Z.^2);
slice(X, Y, Z, W, [-1 1 2], 1, 1)
```

结果如图 9.44 所示。

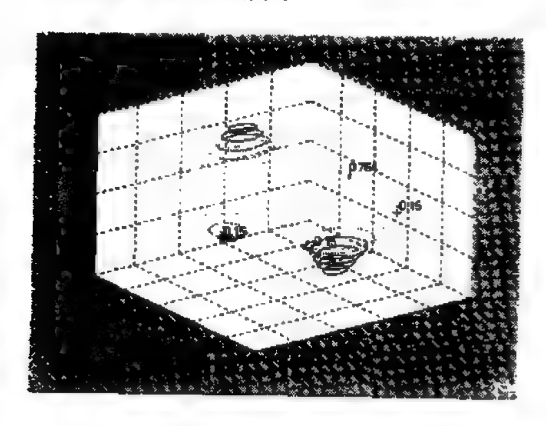




图 9.43　peaks 的空间等高线

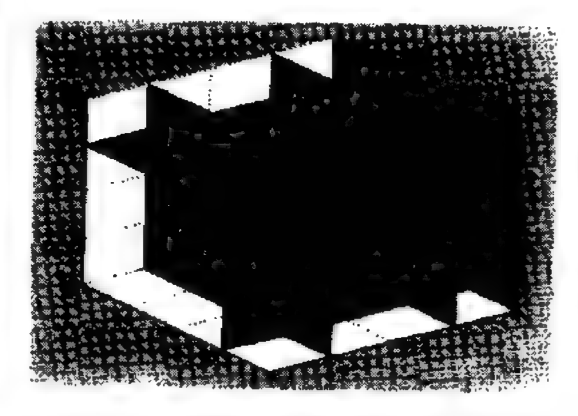


图 9.44　四维空间的表达

9.5.2　在极坐标、柱坐标和球坐标下绘制函数的二维、三维图形

MatLab 提供了直接在极坐标下的绘图命令 ploar 和间接在柱坐标和球坐标下绘制函数图形的命令 pol2cart，sph2cart。

- 在极坐标下绘制函数图形。

polar(theta，rho)命令可以直接将在极坐标系下表达的坐标值矩阵像在 plot 命令中一样绘制成连线图。而且同样可以用'linestyle'字符来控制连线图的线型。

例 39 绘制基圆半径为 1 的渐开线

```
rho0=1;
theta=0: pi/20: 4*pi;
rho=rho0+theta*rho0;
polar(theta, rho, ': ')
```

结果如图 9.45 所示。

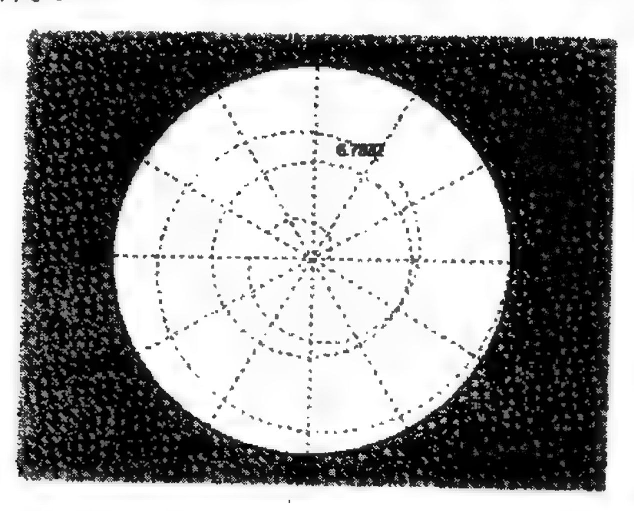


图 9.45 渐开线

- 在柱坐标下绘制函数图形

在 MatLab 中没有在柱坐标系下直接绘制数据图形的命令，但设计了一个能将柱坐标值转换为直角坐标系下坐标值的命令 pol2cart。其使用格式为：

```
[X, Y] = pol2cart(theta, rho)
[X, Y, Z] = pol2cart(theta, rho, z)
```

将柱坐标值转换为直角坐标系坐标值后，就可以用 plot3，mesh，surf 等命令绘图了。

例 40

```
theta=0: pi/20: 2*pi;
rho=sin(theta);
[t, r]=meshgrid(theta, rho);
z=r.*t;
[X, Y, Z]=pol2cart(t, r, z);
mesh(X, Y, Z)
```

结果如图 9.46 所示。

例 41 试将上面极坐标中的例子用本命令改绘出。

- 在球坐标下绘制函数图形

与柱坐标一样，在 MatLab 中也没有直接绘制属于极坐标系下数据图形的命令，但也是设计了一个能将极坐标值转换为直角坐标系下坐标值的命令 sphl2cart。其使用格式为：

```
[X, Y, Z]=sph2cart(theta, phi, rho)
```

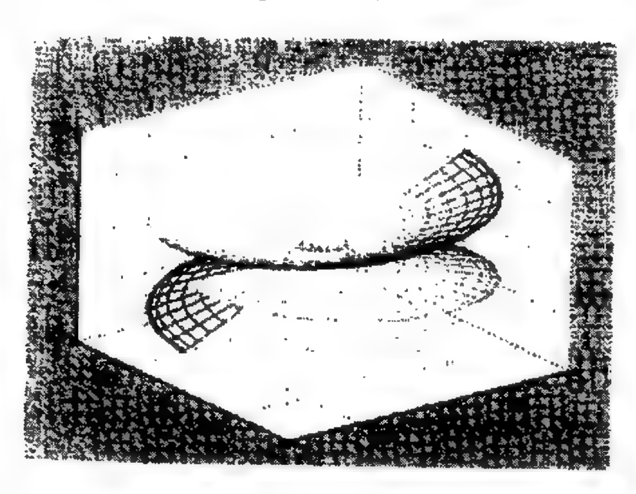


图 9.46　柱坐标绘图

当将柱坐标值转换为直角坐标系坐标值后，同样用 plot3，mesh，surf 等命令进行绘图。
例 42

```
theta=0: pi/20: 6*pi;
phi=theta.^2-theta;
[t, p]=meshgrid(theta, phi);
r=t.*p;
[X, Y, Z]=pol2cart(t, p, r);
mesh(X, Y, Z)
```

结果如图 9.47 所示。

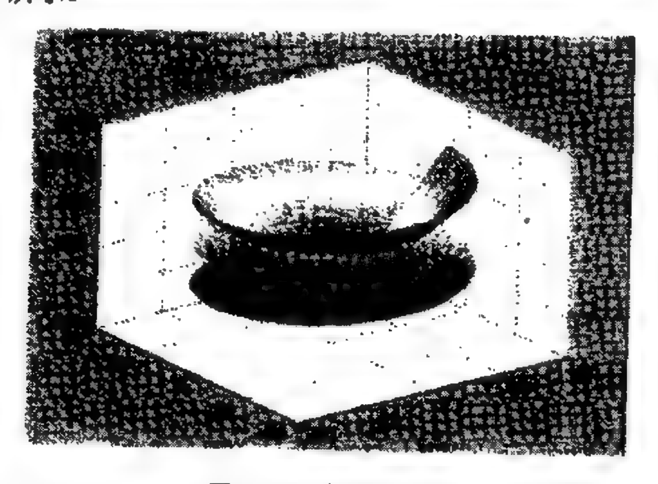


图 9.47　球坐标绘图

附：3 个命令都存在其逆命令，即将在直角坐标系下的坐标值，分别转换为极坐标系、柱坐标系和球坐标系下的坐标值。其命令名及使用格式为：

```
[theta, rho, z]=car2pol(x, y, z)
```

将直角坐标系下的坐标值转换为极坐标系和柱坐标系下的坐标值。

```
[theta, phi, rho]=car2sph(x, y, z)
```

将直角坐标系下的坐标值转换为球坐标系下的坐标值。

9.6 其他格式图形的读取和表现

在 MatLab 中，不仅可以用绘图命令画出图形，还可以直接从其他图像文件中读取图形并将其表现出来。这对其处理其他格式的数据图像提供了方便。

9.6.1 读取图形文件命令 imread

该命令用于读取多种存储格式的图形文件，其返回值有两个：矩阵 x 记录了用于在 MatLab 中画出该图的数据；map 记录了用于显示图形的色图。

其使用格式为：

```
x=imread(filename, fmt)
[X, map]=imread(filename, fmt)
[...] = imread(..., idx)  (TIFF only)
[...] = imread(..., ref)  (HDF only)
```

其中参数 filename 为所读取的文件名，fmt 为其存储格式。两者均要求是以字符串表示。而且所读文件必须在 MatLab 的 path 所包含的文件夹中。

MatLab 可以读取并表现出来的各种图像存储格式如表 9.6 所示。

表 9.6 参数 fmt 允许的文件格式

文件格式	文件类型
'bmp'	Windows Bitmaps (BMP)
'hdf'	Hierarchical Data Format (HDF)
'jpg' or 'jpeg'	Joint Photographic Experts Group (JPEG)
'pcx'	Windows Paintbrush (PCX)
'tif' or 'tiff'	Tagged Image File Format (TIFF)
'xwd'	X Windows Dump (XWD)

9.6.2 绘制所读图形命令 image

该命令的使用格式为：

```
image(x)
```

其作用是将有 imread 命令得到的图像矩阵 x 在 MatLab 中表现出来。一般情况下，不需在使用前说明其色图矩阵 map。

例 43

```
[x, map]=imread('Imgba100', 'jpg');
```

```
colormap(map)
image(x)
```

结果如图 9.48 所示。

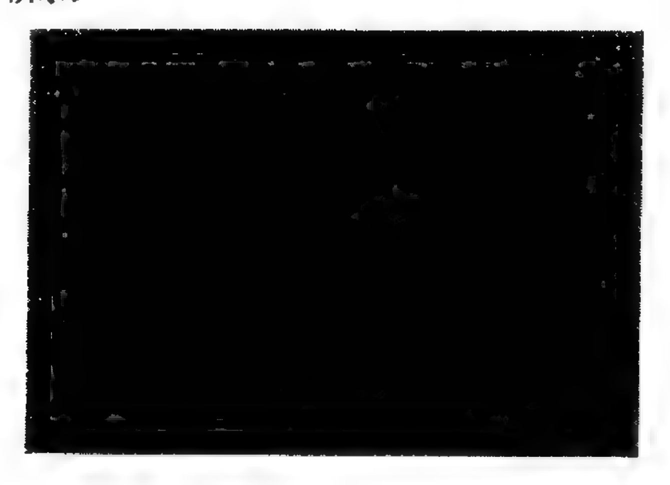


图 9.48　用 imread 命令在 MatLab 表现图像文件 imgba100.jpg

9.7　坐标轴的控制

在 MatLab 中，可以通过对各种参数的设置来完成对坐标轴的控制。其中的高级控制命令涉及到图形句柄，将在 9.10 节中进行专门的介绍。本节只对其初级控制命令 axis 说明。

axis 命令用于控制坐标轴的显示、刻度、长度等特征，其使用格式为：

```
axis([xmin xmax ymin ymax])
axis([xmin xmax ymin ymax zmin zmax])
```

上面两种命令格式分别用于设定二维和三维图形的坐标轴的取值范围。

```
v = axis
```

该命令将返回一个向量 v。且当坐标轴为二维时，v= [xmin xmax ymin ymax]；当坐标轴为三维时，v=[xmin xmax ymin ymax zmin zmax]。

```
axis auto
```

此命令将在考虑图形的基础上，将坐标轴的取值范围设为默认值。该命令也可针对某个具体坐标轴使用，如：

auto z　　　只将 x 轴的取值范围设为默认值。

auto xy　　　将 x，y 两坐标轴的取值范围设为默认值。

axis manual　该命令将冻结当前的坐标轴，以保持坐标轴的取值范围保持不变，即此后在使用 hold on 命令再次在其图形窗口中绘图时，该窗口将保持其坐标轴的取值范围。

axis tight

在考虑所绘图形的基础上，此命令用于将在 3 个方向上的纵高比设为同一个值。

axis fill

该命令用于将坐标轴的取值范围[xmin xmax ymin ymax zmin zmax]分别设为绘图所用数据在相应方向上的最大、最小值。

axis ij

此命令将二维图形的坐标原点设置在图形窗口的左上角。坐标轴 i 垂直向下，坐标轴 j 水平向右。

axis xy

此命令把二维图形的坐标轴恢复为系统的默认状态，即笛卡尔坐标系的设置状态。

axis equal

该命令将使坐标轴在 3 个方向上的刻度增量相同。

axis image

该命令的作用与 axis equal 基本相同，特别地，在 plot 命令建立的坐标轴中，该命令将起 axis tight 的作用。

axis square

此命令将使坐标轴在 3 个方向上长度相等。

axis vis3d

该命令将冻结坐标系此时的状态，以便进行旋转。

axis normal

此命令将撤销命令 axis equal 和 axis vis3d 对坐标轴的操作。

axis off

此命令用于使坐标轴消隐。

axis on

此命令用于恢复消隐的坐标轴。

[mode，visibility，direction] = axis('state')

该命令将返回当前坐标轴的 3 个状态参数：mode， visibility， dirextiton。下面是可能的取值：

```
mode                          'auto' | 'manual'
visibility                    'on' | 'off'
direction                     'xy' | 'ij'
```

9.8 图形标注

与坐标轴的控制命令相似，在 MatLab 中有许多用于控制图形标注的命令和参数。在此介绍其基本的标注命令，关于这方面的高级操作，将在图形句柄部分说明。

9.8.1 标注轴名称和图形标题

xlabel, ylabel, zlabel 是分别用于对 x, y, z 轴进行标注的命令。下面以 xlabel 命令为例，说明其使用格式：

```
xlabel('string')
xlabel(fname)
```

式中 string 是标注在 x 轴上的说明语句，fname 是一个函数名，在此，系统要求该函数必须返回一个字符串供 xlabel 命令进行标注。

Title 命令是标注当前窗口中图形名称的命令，其使用方法和格式与 xlabel 等命令完全相同，如下所示：

```
title('string')
title(fname)
```

9.8.2 标注图形

text, gtext 两个命令用于标注所绘得的具体图形，它们用于对绘得图形的局部性质进行说明。

text 命令的使用格式为：

```
text(x, y, 'string')
text(x, y, z, 'string')
```

其中前者用于二维图形的标注，后者适用于三维图形的标注，其中 x, y, z 是标注语句所处的位置。当 x, y, z 为向量时，将在由这 3 个向量决定的每一点上标注'string'，如果'string'也是与 x, y, z 同维的字符串向量，则在每一点上标注相应的字符串。

Gtext 命令是用于二维图形标注的命令，它可以用鼠标点取标注位置。使用格式为：

```
gtext('string')
```

当程序执行到该命令时，系统的当前图形窗口将被激活，鼠标在此窗口上将显示为一个小十字形，提示选取标注位置，可以在选定位置处单击鼠标进行确定。

9.8.3 图例的标注

当在一幅图中出现多种曲线时，结合在绘制时的不同线性和颜色等特点，用户可以用 legend 命令对不同的图例进行说明。其使用格式为：

```
legend('string1', 'string2', ...)
```

式中'string1', 'string2', …是对图中不同曲线的说明。

9.8.4 控制分格线

命令 grid 是用于控制分格线在所绘图形中是否显示的一个开关命令，其使用格式为：

grid on　在图中使用分格线。

grid off　在图中消隐分格线。

9.8.1　标注轴名称和图形标题

xlabel，ylabel，zlabel 是分别用于对 x，y，z 轴进行标注的命令。下面以 xlabel 命令为例，说明其使用格式：

```
xlabel('string')
xlabel(fname)
```

式中 string 是标注在 x 轴上的说明语句，fname 是一个函数名，在此，系统要求该函数必须返回一个字符串供 xlabel 命令进行标注。

Title 命令是标注当前窗口中图形名称的命令，其使用方法和格式与 xlabel 等命令完全相同，如下所示：

```
title('string')
title(fname)
```

9.8.2　标注图形

text，gtext 两个命令用于标注所绘得的具体图形，它们用于对绘得图形的局部性质进行说明。

text 命令的使用格式为：

```
text(x, y, 'string')
text(x, y, z, 'string')
```

其中前者用于二维图形的标注，后者适用于三维图形的标注，其中 x，y，z 是标注语句所处的位置。当 x，y，z 为向量时，将在由这 3 个向量决定的每一点上标注'string'，如果'string'也是与 x，y，z 同维的字符串向量，则在每一点上标注相应的字符串。

Gtext 命令是用于二维图形标注的命令，它可以用鼠标点取标注位置。使用格式为：

```
gtext('string')
```

当程序执行到该命令时，系统的当前图形窗口将被激活，鼠标在此窗口上将显示为一个小十字形，提示选取标注位置，可以在选定位置处单击鼠标进行确定。

9.8.3　图例的标注

当在一幅图中出现多种曲线时，结合在绘制时的不同线性和颜色等特点，用户可以用 legend 命令对不同的图例进行说明。其使用格式为：

```
legend('string1', 'string2', ...)
```

式中'string1'，'string2'，…是对图中不同曲线的说明。

9.8.4　控制分格线

命令 grid 是用于控制分格线在所绘图形中是否显示的一个开关命令，其使用格式为：

grid on　在图中使用分格线。

grid off　在图中消隐分格线。

```
y1=sin(x);
y2=x./2;
y3=cos(x./0.7+1);
plot(x, y1, 'r-+', x, y2, 'g-.p', x, y3, 'b--o')

title('Three Functions')
xlabel('x')
ylabel('y')
axis([-0.5 4.5 -1.5 2.5])
legend('sin(x)', 'x/2', 'cos(x./0.7+1)')
gtext('Notice!')
```

结果如图 9.50 所示。

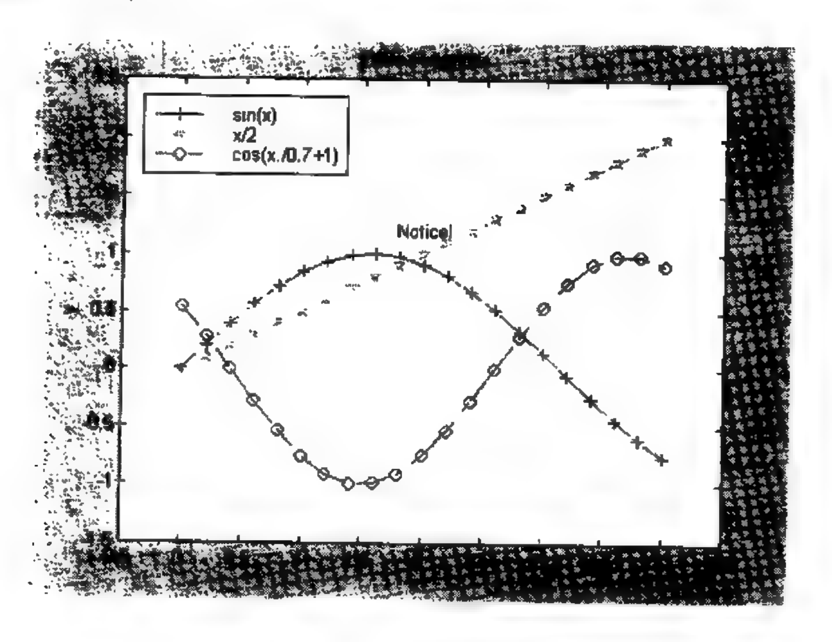


图 9.50　不同函数的表达

9.9　色彩控制

丰富的颜色变化能让图形更具表现力，因此在绘图过程中对色彩的控制和表现是很重要的。在 MatLab 中，色图 colormap 是完成这方面工作的主要命令。

在介绍色图命令之前，先介绍一下各种颜色的产生：在计算机中是用三原色：红(Red)、绿(Green)、蓝(Blue)，通过调整它们的比例来得到需要的颜色。这种方法在 MatLab 中，就表达为一个 1×3 的向量[R G B]，其中 R，G，B 值的大小分别代表这 3 种颜色之间的相对亮度，因此它们的取值范围均必须在[0，1]区间内。通过调整这 3 个量的大小可以得到不同的颜色，一些典型的颜色配比方案如表 9.7 所示。

表 9.7 典型的色极配比方案

原色			调得颜色
红(R)	绿(G)	蓝(B)	
1	1	1	白色(White)
0.5	0.5	0.5	灰色(Gray)
0	0	0	黑色(Black)
1	0	0	红色(Red)
0	1	0	绿色(Green)
0	0	1	蓝色(Blue)
1	1	0	黄色(Yellow)
1	0	1	洋红色(Magenta)
0	1	1	青色(Cyan)
0.5	0	0	暗红色(Dark Red)
1	0.62	0.4	红铜色(Copper)
0.49	1	0.83	碧绿色(Aquamarine)

有了调好的颜色，就可以用此颜色作为绘图用色。其设定命令为：

```
colormap([R, G, B])
```

例 46

```
colormap([1 0.62 0.4])
mesh(peaks(80))
```

结果如图 9.51 所示。

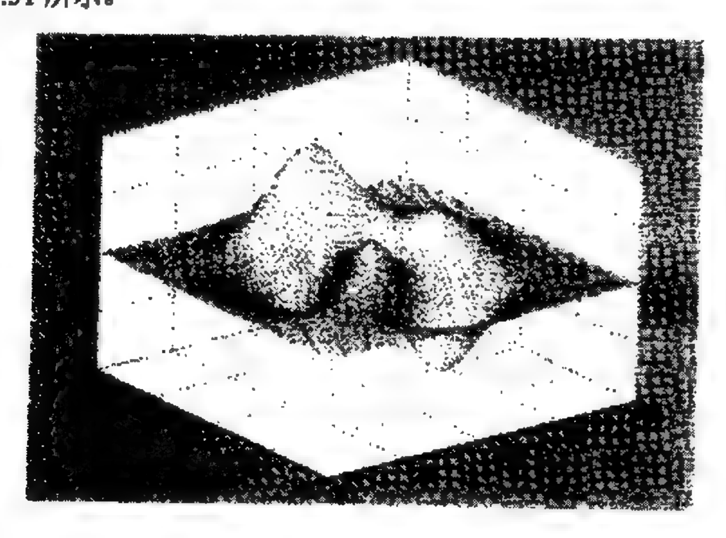


图 9.51 红铜色的 peaks 图像

在例 9.46 中，由于指定的绘图用色只有一种，因此只能画出单色图形。如果需要用自己设置的多种颜色绘出图形，就要将上面的颜色向量扩展为一个 m×3 的颜色矩阵，这个矩阵在 MatLab 中称色图。色图中每行元素的含义与元素向量中的相同，因此其取值范围也必须限制在[0，1]之间。

当确定了所需色图矩阵 T 后，可用下面的命令将其设为绘图所用的色图。

```
colormap(T)
```

为了方便×使用，MatLab 提供了几种典型的常用色图，关于这些色图的名称和其产生函数如表 9.8 所示。

表 9.8 色图名称及产生函数

色图名称	产生函数
蓝色调灰度色图	bone
青红浓淡色图	cool
线性纯铜色图	copper
红白蓝黑交错色图	flag
线性灰度色图	gray
黑红黄白色图	hot
饱和色图	hsv
一种色图的变体	jet
粉红色图	pink
光谱色图	prism

其中，在普通绘图中的默认色图是 hsv。

有两种设置色图的格式如下：

hsv 色图矩阵为 64×3。

hsv(n) 色图矩阵为 n×3。

例 47

```
x=0: pi/10: pi;
T=[abs(sin(x).^2); abs(cos(x)); abs(sin(2.*x))]';
colormap(T)
surfl(peaks(80))
figure
colormap(copper)
surfl(peaks(80))
```

结果如图 9.52，9.53 所示。

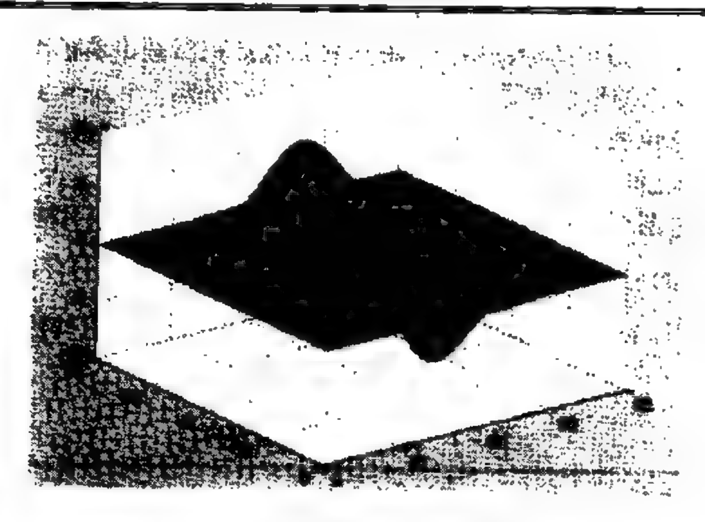


图 9.52 线性纯铜色图

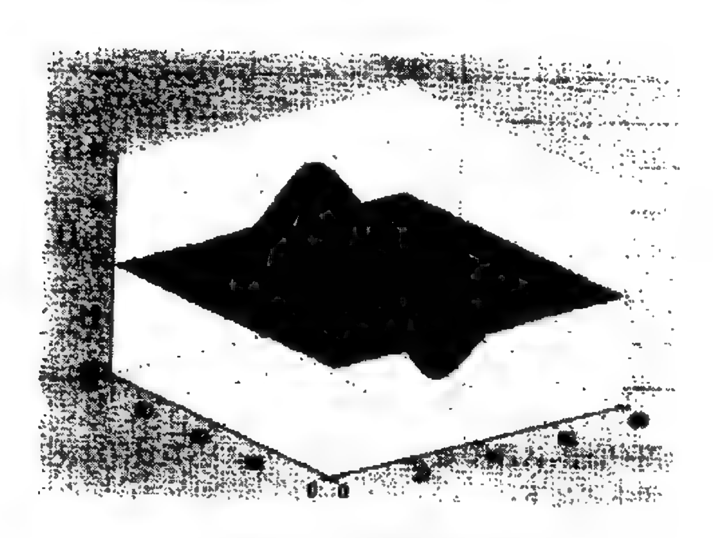


图 9.53 自定义色图

9.10 高级图像处理

在 MatLab 中，图形句柄是系统的底层图形控制系统。在这个控制系统中，有些命令专门用来对图形进行更细腻的处理，在实际计算中有着广泛的应用。

9.10.1 图形对象

在前面介绍的图形处理命令中，都是以图形窗口内的全部图形为操作对象。实际上，可以针对其中某一个部分进行单独处理，就是对图形窗口中的图形对象进行处理。在 MatLab 中，图形对象包括：根屏幕(ROOT SCREEN)、图形窗口(FIGURE)、轴(AXES)、线(LINE)、块(PATCHES)、面(SURFACES)、像(IMAGE)、字(TEXT)、用户界面控制(USER

INTERFACE CONTROLS)和用户界面菜单(USER INTERFACE MENUS)。

下面对这些对象加以简单介绍。

- 根屏幕　指向计算机屏幕。每个 MatLab 系统内只允许有一个根屏幕，而且后面的图形对象将都在根屏幕的基础上产生，在这个意义上，根屏幕的辈分最高。
- 图形窗口　在根屏幕基础上创建的窗口，它的数目不限。除了根屏幕外，其他的图形对象都要在图形窗口的基础上产生，在这个意义上，如果根屏幕是祖父辈，则图形窗口就是父辈。
- 轴　它将在图形窗口内创建一个区域并为创建线、块、面、像和字作准备。所有绘图命令都会产生一个轴对象，可以用 axes 命令创建。它在层次上相当于孙辈。
- 线　用来创建图形的一个基本对象，可以用绘制连线图的命令创建。线为轴的子辈，在层次上相当于重孙辈。
- 块　用来填色多边形。填充命令 fill、fill3 可以创建此种对象。与线一样为对象层次上的重孙辈。
- 面　是矩阵数据的空间表现，可以用创建面命令来创建。与线辈分相同。
- 像　为数值矩阵及其色图的表现。由 image 命令创建。与线辈分相同。
- 字　即字符串。可用 title、xlabel、gtext 等命令创建。与线辈分相同。
- 用户界面控制　允许用户使用鼠标完成一定的功能。为图形窗口的子辈。
- 用户界面菜单　允许用户在图形窗口建立菜单。为图形窗口的子辈。

9.10.2 图形对象的句柄和创建图形对象

- 句柄

句柄的含义很简单，就是依附于图形对象并用于区别不同图形对象的一个标志。如 C，Foutran 等计算机语言，图形对象就如系统的一个输出文件，而在进行输出时一定要给出一个通道号用以标志输出文件，在 MatLab 中，句柄就是这个通道号。

根屏幕的句柄值总为零，图形窗口的句柄为一整数，其他图形对象的句柄值则是实数。只要使用图形命令时，系统都将给出新创建的图形对象的句柄值。

- 创建图形对象

从图形对象的分类可知，系统已经将图形分解得很细。其目的就是完成对这些细节的单独操作，而不像其他高级命令对很多对象都不可由用户操作。

在 MatLab 的所有图形对象中，除了根屏幕不必由系统创建外，每个对象都各由一个内部函数来单独创建。这些命令名的列表和使用格式如表 9.9 所示。

表 9.9　创建图形对象的命令名、功能及使用格式

创建图形对象函数名	功　能	使用格式
figure	创建图形对象	H=figure(n)(n 必须为整数)
axes	创建轴对象	H=axex('position', [left, b-ottom, width, height]) 设定坐标盒的位置和尺寸
line	划线	H=line(x, y, z) 绘出由 x, y, z 确定的直线

续表

创建图形对象函数名	功　能	使用格式
patch	创建并填充多边形	H=patch(x，y，z，c) 由 x，y，z 确定多边形，c 控制填充用色
surface	创建空间曲面	H=surface(x，y，z，c) 由 x，y，z 确定空间曲面，矩阵 c 控制色彩
image	创建并显示图像对象	H=image(X)
text	创建字体对象	H=text(x，y，z，'string')
uicontrol	用户界面控制	H=uicontrol('property'，value)
uimenu	用户菜单控制	H=uimenu('property'，value)

上面这些命令都有一个返回值 H，该值即为所创建图形对象的句柄值。

下面是这些命令的几个具体使用实例。

例 48

```
x=1: 10;
y=x.^2/10;
z=y.*x;
h1=figure(1)

h2=subplot(2, 2, 1)
h3=line(x, y, z)

h4=subplot(2, 2, 2)
h5=patch(x, y, z)

h6=subplot(2, 2, 3)
[x, y]=meshgrid(x, y);
z=y.*x;
h7=surface(x, y, z)
h8=subplot(2, 2, 4)
h9=text(0.1, 0.5, 0, 'Can you see me ?')
```

程序执行结果为：

```
h1 =
                        1
h2 =
          1.0015869140625
h3 =
          2.0003662109375
h4 =
          3.0006103515625
h5 =
          4.0006103515625
```

```
h6 =
            5.0008544921875
h7 =
            6.0008544921875
h8 =
            7.0008544921875
h9 =
            8.0008544921875
```

绘得图如图 9.54 所示。

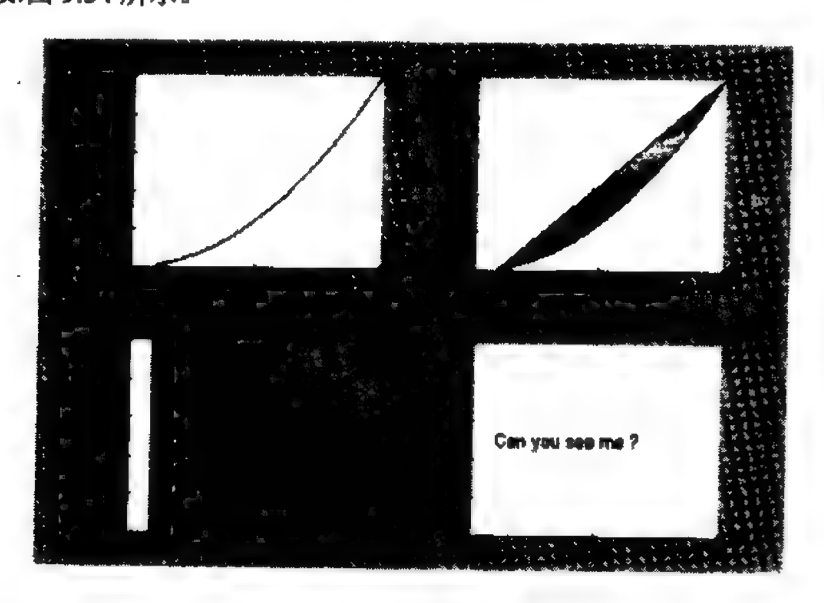


图 9.54　创建句柄图形

9.10.3　图形对象句柄的查询与删除

在 MatLab 中，查询当前图形窗口的句柄值和当前轴的句柄值的命令分别为：

```
h=gcf
h=gca
```

其删除命令为：

```
delete(h)
```

该命令将删除 h 连接的图形对象及其所有子辈、孙辈图形对象。

9.10.4　图形对象的性质

图形对象的性质决定了图形的显示。通过调整对象性质，可以完成对图像更细腻的处理。

- 共性

从 MatLab 中图形对象的辈分层次上看，它与 C++，Java 语言的类很相似。在 C++和 Java 语言中，子类将继承父类的属性，而在 MatLab 中，辈分低的对象具有前辈的主要性质。因此，在图形对象中也就形成了许多各个对象均具备的性质——即图形对象的共性。

这些共性如表 9.10 所示。

表 9.10 图形对象的共性

共性名称	含 义
BusyAction	控制中断方法
Children	子辈
CreateFcn	对象创建后回应系统
DeleteFcn	对象删除后回应系统
HandleVisibility	授权用户可以通过命令控制句柄的有效性
Interruptible	决定回应程序是否可以中断
Parent	父辈
Selected	标志对象是否被选中
Tag	用户定义的对象句柄
Type	对象类型(Figure, Line, Text 等)
UserData	与图形对象相关的数据
Visible	决定对象是否可视
Property	包含性质

- 特性

除了对象都具有的共性，每一种图形对象都还具有自己的特性，可以通过对这些特性的设置完成深入的图形图像处理。各个图形对象的特性名称及特性值如表 9.11～9.19 所示。

表 9.11 根屏幕特性

特性名称	含 义	特 性 值
Clipping	图形剪辑是否打开	on, off
CurrentFigure	当前图形窗口	当前图形窗口的句柄值
Dirary	日记是否打开	on, off
Echo	回显是否启动	on, off
Format	数据输出格式	short, long, ...
FormatSpacing	输出空格格式	compact, loose
Interruptible	中断允许	yes, no
Visible	根屏幕是否可视	on, off

表 9.12 图形窗口的特性

特性名称	含 义	特 性 值
Color	设置窗口颜色	[R G B]
Colormap	设置色图	色图矩阵
CurrentAxes	当前轴	该轴句柄
CurrentObject	当前对象	当前对象的句柄

续表

特性名称	含 义	特 性 值
Name	图形窗口的标题	字符串
InverHardcopy	图形硬拷贝时颜色是否反显	on，off
MenuBar	菜单条开关	on，off
NextPlot	设定下个绘图方式	add，new，replace
PaperOrientation	横向或竖向打印	Portrait，landscape
PaperPosition	打印用纸张位置	通常取默认值：[0.25 2.5 8.0 6.0]
PaperSize	纸张大小	[x，y] x，y为纸张右上角的坐标值
PaperUnits	计算纸张大小的单位	Iinches，centimeters...
Pointer	鼠标样式	Aarrow，crosshair，...
ScreenSize	图形窗口的大小位置	[left，bottom，width，height]
Resize	设定窗口大小是否可变	on，off
Sharecolors	公共颜色	yes，no
Units	绘图时的计量单位	Ppoxels，inches，...

表9.13 轴的特性

特性名称	含 义	特 性 值
AspectRatio	视图比例	[纵横轴比，刻度比]
Box	坐标轴是否封闭	on，off
CLim	色轴取值范围	[min，max]
CLimMode	设定色轴取值范围的方法	auto，manul
Color	坐标轴定义区的颜色	none，ColorSpec
ColorOrder	画连线图的用色顺序	颜色矩阵
DrawMode	设定绘图速度	normal，fast
FontName	字体名称	Helvetica
FontSize	字体大小	[整数]
GridLineStyle	分格线线型	-，--，:，-.，...
LineWidth	线宽	实数
Position	设定轴的位置	[left，bottom，width，height]
TickLength	轴上刻度长度	[二维用长度，三维用长度]
TickDir	设定刻度方向	In，out
Units	单位	normalized，pixel，inches，...
View	设定视角	[az，el]
XGrid	是否绘出x方向的分格线	on，off
XLabel	X轴的标注	"string"的句柄
XScale	X轴的刻度方法	linear，log

表 9.14 线的特性

特性名称	含 义	特 性 值
Color	线的颜色	ColorSpec，RGB
EraseMode	擦除方法	normal，none，backgound
LineStyle	线型	-，--，:，-.-，…
LineWidth	线宽	实数
Marker	设定标记	O，x，+，…
MarkerSize	标记大小	实数，默认 6
MarkerFaceColor	标记内部填充颜色	auto，none，[R G B]

表 9.15 面的特性

特性名称	含 义	特 性 值
Cdata	设定 Zdata 色彩	数值矩阵
EdgeColor	设定网格线的颜色	ColorSpec，none，flat，interp
EraseMode	擦除方法	normal，none，backgound
FaceColor	设定网格线的表面颜色	ColorSpec，none，flat，interp
LineStyle	线型	-，--，:，-.-，…
LineWidth	线宽	实数
MarkerSize	标记大小	实数，默认 6
MeshStyle	设定经纬格线模式	both，row，column

表 9.16 块的性质

特性名称	含 义	特 性 值
Cdata	设定块的边界颜色	数值向量
EdgeColor	设定网格线的颜色	ColorSpec，none，flat，interp
EraseMode	擦除方法	normal，none，backgound
FaceColor	设定网格线的表面颜色	ColorSpec，none，flat，interp

表 9.17 字的特性

特性名称	含 义	特 性 值
Color	设定字的颜色	ColorSpec(v)
FontName	字体名称	Helvetica
FontSize	字体大小	[整数]
FontUnderline	字符串是否加下划线	on，off
HorizontalAlignment	字与选定点的水平关系	left，center，right
Rotation	旋转角度	+-(270，180，90，0)
VerticalAlignment	字与选定点的垂直关系	top，cap，middle，baseline，bottom

表 9.18　用户界面控制的特性

特性名称	含　义	特 性 值
BackGroundColor	控制按钮的背景色	ColorSpec
CallBack	设定控制按钮的功能	字符串
ForeGroundColor	设定控制按钮的字符颜色	ColorSpec
HorizontalAlignment	设定控制按钮名称的位置	left，center，right
Position	设定控制按钮在图形窗口中的位置	[left，bottom，width，height]
String	设定控制按钮名称	以|分隔的字符串组
Style	设定控制按钮的类型	pushbutton，radiobutton，checkbox，...
Units	设定计量单位	pixel，normalized，inches，...

表 9.19　用户界面菜单的特性

特性名称	含　义	特 性 值
Accelerator	快捷键设定	Ctrl+字母
CallBack	调用功能设定	字符串
Enable	现场允许使用变色	on，off
Label	菜单命名	字符串
Position	菜单条位置	实数
Separator	菜单项划线模式设定	on，off

9.10.5　图形对象性质的设置

可以使用两种方法设置图形对象的性质：

- 在创建对象的同时设置其性质

此种方法要求使用专用创建函数进行，其设置格式为：

```
    h=function(..., 'PropertyName1', PropertyValue1, 'PropertyName2',
PropertyValue2, ...)
```

例 49

```
    t=0: 0.5: 5;
    x=0: 0.5: 5;
    F=sin(x).*exp(x);
    momentum=F.*x;
    h1=figure('Color', [0.49 1 0.83], 'Pointer', 'cross')
    h2=axes('Box', 'on', 'TickDir', 'out', 'AspectRatio', [1.5, nan],
'DrawMode', 'fast', 'LineWidth', 2, 'XGrid', 'on', 'YGrid', 'on')
    h3=line(x, F, momentum, 'LineStyle', '-', 'LineWidth', 2, 'Color', [1 0
0], 'Marker', 'o', 'MarkerSize', 10, 'MarkerFaceColor', [0 1 1])
    h4=text(3.5, 0, 'TheEighthPoint', 'FontSize', 15, 'FontUnderline', 'on',
'Rotation', -40)
```

结果如图 9.55 所示。

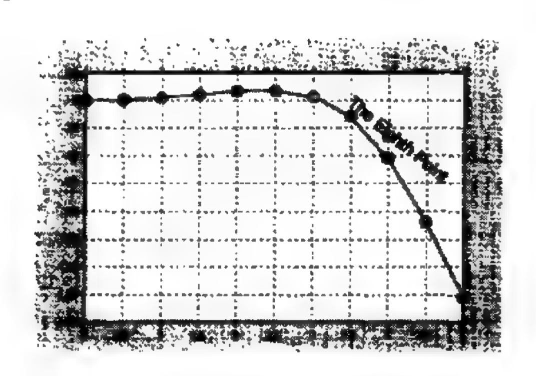


图 9.55　设置对象特性后的图形

且各句柄值如下：

```
h1 =
          2
h2 =
          4.0009765625
h3 =
          5.000732421875
h4 =
          6.000732421875
```

例 50

```
y=peaks(20);
h1=figure('Color', [1 0.62 0.5], 'Name', 'Peaks')
h2=axes('Box', 'on', 'TickDir', 'in', 'DrawMode', 'fast', 'XGrid', 'on',
'YGrid', 'on', 'ZGrid', 'on', 'View', [45, 45])

h3=surface(y, 'LineWidth', 1.5, 'EdgeColor', [0.5 0.6 0.7], 'FaceColor',
'interp', 'MarkerEdgeColor', [0.2 0.5 0.7], 'MarkerFaceColor', [0 1 1],
'Marker', 'o', 'MarkerSize', 5)

h4=text(10, 15, 10, 'TOP', 'FontSize', 15)
```

该程序执行后，各句柄值为：

```
h1 =
          1
h2 =
          1.0029296875
h3 =
          2.0010986328125
h4 =
          3.0010986328125
```

绘得结果如图 9.56 所示。

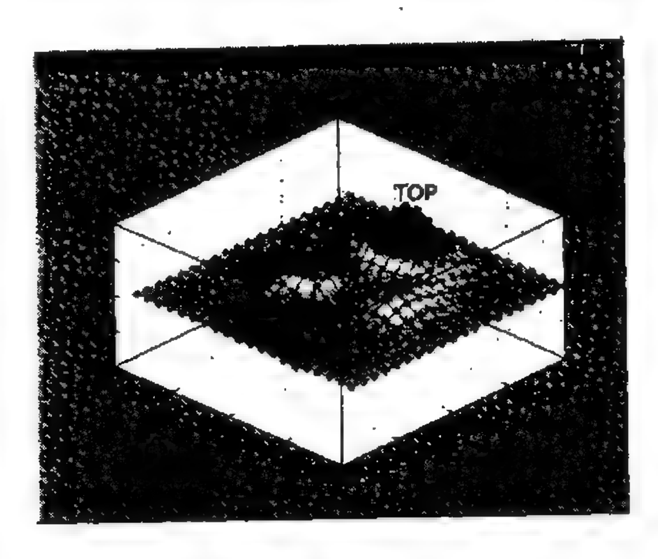


图9.56 设置对象性质后的peaks的图形

- 用set命令设置已创建对象的性质

该命令可以对已创建的对象性质通过句柄进行修改。其使用格式为：

```
set(h, 'PropertyName', Property_Value)
set(h , 'PropertyName1' , Property_Value1 , 'PropertyName2' ,
Property_Value2, ...)
```

现重新对上例进行设置：

```
set(h1, 'Color', [0.7 0.5 0.6])
set(h2, 'Box', 'off', 'TickDir', 'out')
set(h3, 'FaceColor', 'none')
```

结果如图9.57所示。

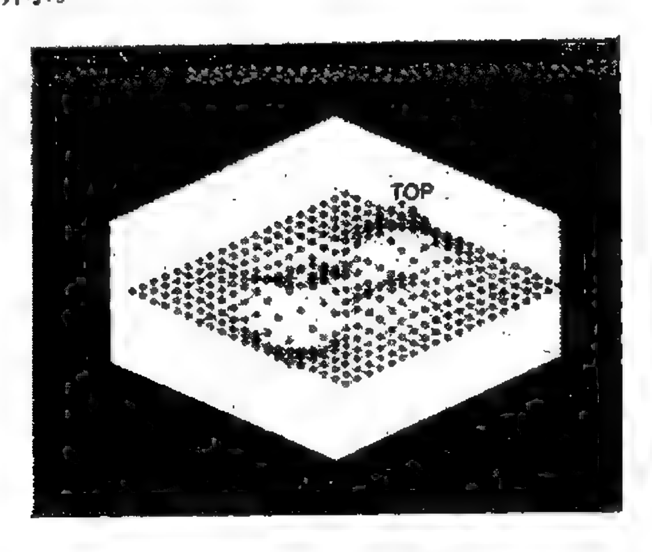


图9.57 set修改后的图形

9.10.6 图形对象性质的查询

在 MatLab 中，可以用 get 命令查询对象性质，其使用格式为：

```
get(h)
get(h, 'PropertyName')
```

如图 9.57 为例进行查询：

```
get(h3)
```

查询结果为：

```
CData = [ (20 by 20) double array]
CDataMapping = scaled
EdgeColor = [0.5 0.6 0.7]
EraseMode = normal
FaceColor = interp
LineStyle = -
LineWidth = [1.5]
Marker = o
MarkerEdgeColor = [0.2 0.5 0.7]
MarkerFaceColor = [0 1 1]
MarkerSize = [5]
MeshStyle = both
XData = [ (1 by 20) double array]
YData = [ (20 by 1) double array]
ZData = [ (20 by 20) double array]
FaceLighting = flat
EdgeLighting = none
BackFaceLighting = reverselit
AmbientStrength = [0.3]
DiffuseStrength = [0.6]
SpecularStrength = [0.9]
SpecularExponent = [10]
SpecularColorReflectance = [1]
VertexNormals = [ (20 by 20 by 3) double array]
NormalMode = auto

ButtonDownFcn =
Children = []
Clipping = on
CreateFcn =
DeleteFcn =
BusyAction = queue
HandleVisibility = on
HitTest = on
Interruptible = on
```

```
Parent = [1.00317]
Selected = off
SelectionHighlight = on
Tag =
Type = surface
UIContextMenu = []
UserData = []
Visible = on

get(h2, 'DrawMode')
```

查询结果为：

```
ans =
fast
```

9.11　动　　画

9.11.1　动画的制作

在 moviein，getframe 命令下，可以完成动态数据到动态画面的制作。其使用格式及制作步骤为：

M=moviein(n)

该命令是创建一个矩阵 M，此矩阵共有 n 列，每一列将存储一帧画面。

M(：，i)=getframe

该命令将当前图形窗口中的画面作为第 i 帧以列的形式存入矩阵 M。

Movie(M，k)

该命令将按列的顺序放映矩阵 M 中存储的画面，并重复 k 次。

例 51

```
x=-3: 0.1: 3;
[x, y]=meshgrid(x);
z=sin(x.*y).*exp(x.*y/5);
mesh(z);
M=moviein(30);
axis manual
set(gca, 'nextplot', 'replacechildren');
for j = 1: 30
    mesh(cos(4*pi*j/30)*z, z)
  M(: , j)=getframe;
end
movie(M, 25)
```

9.11.2 彗星轨线——动态图形的展现

利用彗星轨线命令可以非常容易的绘出质点的运动轨迹，其使用格式为：

```
comet(y)
comet(x, y)
comet(x, y, p)
comet3(z)
comet3(x, y, z)
comet3(x, y, z, p)
```

其中，comet 命令适用于二维平面，comet3 命令适用于三维平面。式中对 x，y 的要求与在 plot，plot3 命令中的要求相同。参数 p 设定绘得的彗星轨线的彗长为 p×length(z)。

例 52

```
z=0: 0.1: 100;
x=sin(z);
y=cos(z).*10;
comet3(x, y, z)
```

绘得结果如图 9.58 所示。

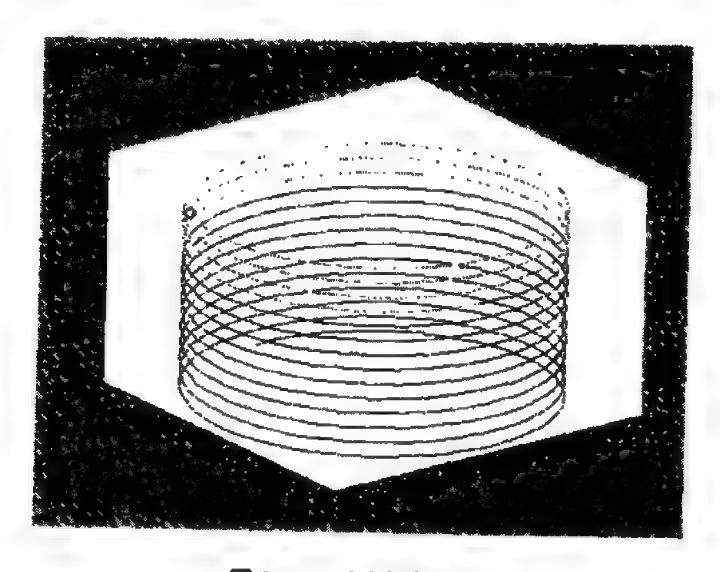


图 9.58 动态螺旋线图

9.12 习 题

(1) 试用 subplot 命令在同一图形输出窗口中绘出如下 3 个函数的图形。

① $y=x^2$ [-2,2]

② $y=x^{1/2}$ [0,4]

③ $y=x^{-1/2}$ [0,4]

(2) 试绘出颜色为洋红色，数据点用五角星标出的函数 $y=xe^{\sin x}$ 在[0,5]上的虚线图。

(3) 绘出函数 $f=x^2+e^y|x|$的在[-2,2], [-2,2]上的三维瀑布图。

(4) 绘出四维函数 $f=x\sin(y/z)$的四维表现图。

(5) 绘出矢量场 $z=x \cdot *y+e^x$ 的空间矢量场图，并对图形进行标注。

(6) 试用线性纯铜色重绘题 3 中函数的表面图。

(7) 完成函数 $z=x^2+y^2\sin x$ 的动画表达。

(8) 绘出函数 $x=\sin t, y=t^2+e^t$ 的彗星效果图。

第 10 章　调　试

MatLab 有一套比较完整的调试命令，再加上其运算命令的调用就十分简单，又有相应的帮助信息，因此，MatLab 程序的调试远比 C，FORTURN 等语言的调试容易得多。

此外，除了调试命令用来检查程序的正确性外，MatLab 在当执行的程序有错时会自动终止运行，并在其命令窗口(MatLab Command Window)中给出相应错误的提示信息。

调试程序中一般有如下几个步骤：

(1) 在程序可疑处设置断点。

(2) 执行程序。

(3) 检查程序运行至断点处的各变量的当前值。

(4) 改变局部工作内存中的前后关系。

(5) 列出调用堆栈。

(6) 恢复执行。

(7) 结束调试状态。

(8) 取消断点。

下面将按所列出的步骤依次介绍其相应的命令。

10.1　设置断点 dbstop

在设置断点之前，首先要知道断点所在行的行号。命令 dbtype M_File 将会在命令窗口列出程序 M_File，并在每行前面加上相应的行号。

dbstop 命令专门用来设置程序停止运行的位置，它的使用格式为：

```
dbstop at LineNo in Function
dbstop in Function
dbstop if keyword
```

第 1 种格式的作用是在程序 Function 的第 LineNo 行处设置断点。

第 2 种格式的作用是确认系统在下面的调试过程中的调试对象是程序 Function。

第 3 种格式的作用是使程序在发生属于由 keyword 描述的错误时终止运行，并给出相应提示信息。其中 keyword 为：

- error

当程序在执行过程中发生运行时间错误时， dbstop if error 将停止并退出程序的运行。

- naninf 和 infnan

当程序在运算过程中出现非数(NAN)或无穷大(inf)时， dbstop if naninf 将停止并退出程序的运行。

- warning

当程序在运行过程中出现运行时间警告时，dbstop if warning 将停止并退出程序的运行。

例 1

调试程序 temp：

```
[X, Y]=meshgrid(-1: 0.1: 1, -1: 0.1: 1);
Z = 1./X.*Y;
mesh(X, Y, Z)
```

具体调试命令如下：

```
dbstop in temp
dbstop if naninf
temp
```

调试结果为如下警告信息：

```
Warning:  Divide by zero.
> In E: \bin\book\temp.m at line 2
K?
```

最后一行的字母 K 是系统处于调试状态的提示符。如果使用的是高版本的 MatLab，系统将自动启动它的编辑/调试窗口(MTALAB Edit/Debugger)，在此窗口内的各种调试按钮也同时可以使用，具体窗口如图 10.1 所示。

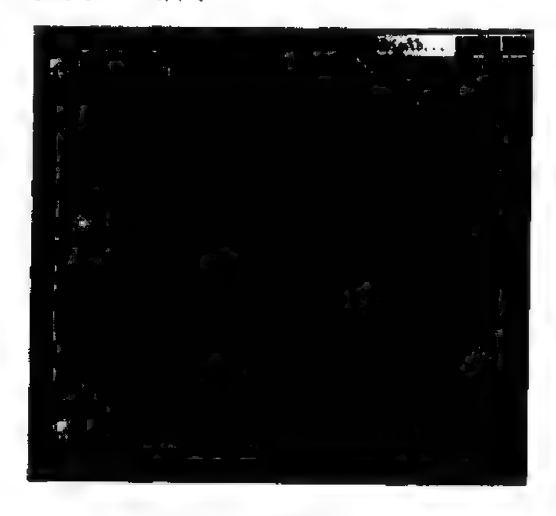


图 10.1　编辑/调试窗口

另外，在编辑调试窗口中的小图标组中的图标也可以在光标所在行处设置断点。

10.2　核查运行过程中变量的当前值

当程序运行至断点处时将自动暂停执行，此时在调试提示符 K 下可以像在平时一样通

过输入变量名来检查该变量的当前值，用 who，whos 命令核查所有变量的总体情况。

最简单的办法是单击命令窗口工具条中的 Workspace Browser 图标命令，它将创建一个新窗口 Workspace Browser，如图 10.2 所示。

可以通过其中的 open 命令在编辑/调试窗口中打开其中的数值型变量并在编辑/调试窗口中进行更改。

以上操作只是最基本的调试命令，还可以在调试状态下进行对局部工作空间内存的前后顺序的改变。其命令为 dbup，dbdown。

在 MatLab 启动之后，系统将自动开辟一工作空间并为其分配相应内存，这一工作空间被称为基本工作空间(base workspace)。当在此空间内执行一个函数时，系统将专门为此函数分配一个子空间和相应的内存。因此，当函数处于执行或调试过程中时，由该函数创建的变量尚处于子空间内而并未传递到基本工作空间中，因而在基本工作空间中是“看”不到这个变量的。



图 10.2 Workspace Browser 窗口

根据上面所说的函数执行过程，可知：当系统进入调试状态时，处于当前活动空间的是函数的子空间，所作的一切改变只是在该子空间内有效。但有时会遇到在调试过程中需要到其基本空间或该子空间的子空间内的情况，这时就要用到 dbup 和 dbdown 命令。

这两个命令，前者的作用是将用户带入上一层空间，后者是将用户带回执行 dbup 命令前的空间。

例 2

函数 pro1：

```
function w=pro1
syms W
x=sym(W);
y=sym(2);
w=x*y;
```

函数 pro2：

```
function z=pro2
x=3;
y=4;
z=x*y;
```

```
z=pro1;
```

输入以下命令：

```
dbstop 4 in pro1                %在程序pro1中第4行设置断点
pro2                            %执行程序pro2
```

由于在程序pro1中第4行中设置了断点，程序将在此处暂停执行并进入调试状态。

例 3

下面所执行的命令用来调换工作空间和检查变量的值：

K?x	
x =	在当前工作空间内查询变量x的值
W	
K?dbup	进入当前空间的上一层工作空间
In workspace belonging to E:	此结果说明此工作空间为函数pro2的工作空间
\bin\book\debug\pro2.m	在函数pro2的工作空间内查询变量x的值
K?x	进入函数pro2的工作空间的上一层工作空间，为MatLab的基
x =3	本空间(base workspace)
	在基本空间内查询变量x的值
K?dbup	
In base workspace	
K?x	下降到函数pro2的工作空间
??? Undefined function or variable 'x'。	
K?dbdown	下降到最初的当前空间，由结果知其为函数pro1的工作空间
In workspace belonging to E:	
\bin\book\debug\pro2.m	
K?dbdown	
In workspace belonging to E:	
\bin\book\debug\pro1.m	

MatLab 除了可以自由控制和选择工作空间，还可以列出其所调用的堆栈。命令为dbstack，这个命令将返回程序从开始执行起直到当前断点处之间调用的所有函数的函数名。

例 4

接着上例，断点不变，在调试状态下输入：

K?dbstack

```
> In E: \bin\book\debug\pro1.m at line 5
  In E: \bin\book\debug\pro2.m at line 5
```

10.3 恢复执行

至此，在一个断点上能作的已经差不多了，下面讲恢复程序的执行。有两个命令：dbstep 和 dbcont。

命令 dbstep 的作用是自当前暂停处向下执行一行。此命令在编辑/调试窗口中的工具条中的图标为：■。

命令 dbcont 的作用是自当前暂停处一直向下执行直至程序结束或遇到另外的断点。此命令在编辑/调试窗口中的工具条中的图标为：■。

10.4 结束调试

命令 dbquit 用于在调试过程中希望中途退出的情况。

dbquit 命令在编辑/调试窗口中的工具条中的图标为：■。

10.5 取消断点

命令 dbclear 适用于需要取消一些或全部断点的情况。其适用格式与 dbstop 类似：

dbclear at Lineno in Function

dbclear all in Function

dbclear all

dbclear in mfilename

第 1 个命令格式用来定点清除断点——在程序 Function 中的第 Lineno 行处的断点。

第 2 个命令格式用来清除程序 Function 中的所有断点。

第 3 个命令格式用来清除在系统目录下的所有 M 文件中的断点。但其中不包括设为 errors 和 warning 的断点。

dbclear 命令在编辑/调试窗口中的工具条中的图标为：■。

这就是 MatLab 的基本调试命令，它具有较大的灵活性，应用心掌握。

第 11 章　MatLab 与其他语言计算功能的连接

MatLab 的强大之处还在于它能跟许多程序语言对话，如 C，FORTRAN 语言和 Microsoft Word 等。在 MatLab 中，可以像调用 MatLab 自己的函数或命令一样调用编译后的 C 或 FORTRAN 的函数，同样，在 C 或 FORTRAN 程序中，也可以调用 MatLab 的函数或命令，使得这些语言可以充分利用 MatLab 强大的矩阵计算和方便的绘图功能。在 MatLab 中，由于循环速度较慢，使得整个程序的执行速度慢而效率低，但 C 和 FORTRAN 的循环较快，通过编写合适的 C 或 FORTRAN 函数让 MatLab 调用，可以很好地解决这一问题。同时，在 C 或 FORTRAN 程序中，可以调用 MatLab 生成的 MAT 文件并读取和写入矩阵数据，实现同 MAT 文件之间的数据交换。可见，MatLab 与 C 或 FORTRAN 语言之间的连接是非常灵活的，灵活地运用可以大大提高工作效率。

11.1　基本概念和函数

MatLab 是解释性语言，对命令是边解释边执行的，对于执行次数比较多的循环，其执行速度较慢，特别是对于多层嵌套循环，就更慢了，因而，MatLab 是不适合用作数值计算的。但是 MatLab 强大的矩阵运算和绘图功能是其他任何语言都无法比拟的，通过调用快速执行的 MEX 文件，便能克服上述缺点，充分利用两者的长处。MatLab 的库函数大都是用 FORTRAN77 和 C 编写的，因而执行速度较快，利用更高版本 FORTRAN 和 C 语言编写的函数也可兼容。一般情况下，并不赞成编写 MEX 文件，MatLab 是在一个较高层次上对矩阵进行运算的，而在 MEX 文件中，并不能体现 MatLab 的这一优势，而使矩阵运算变得更加复杂起来。

MatLab 对 MEX 文件有优先执行的功能，如果同时存在文件名相同的 M 文件和 MEX 文件，则 MatLab 会执行 MEX 文件，而不是 M 文件。MatLab 调用 MEX 文件与调用 M 文件一样，除了在速度上有所差别之外，其他方面没有任何差别。

在 MatLab 中，最常用的二进制文件是 MAT 文件，MatLab 还提供了用以读写 MAT 文件的 C 和 FORTRAN 的程序库，能够以特定的格式读写二进制或文本文件，这在与那些使用特定的数据文件的程序进行数据交换时很有用。

MatLab 是由 C 编写的，其中的数据都是使用指针来调用的，这就要求用 C 和 FORTRAN 编写的 MEX 文件中也要大量的使用指针来调用数据。通常，如果引用一个指针的名字，就相当于引用了指针所指向的对象。在 C 中经常用到指针，但在 FORTRAN 中却没有指针的概念，因而在 FORTERAN 中对矩阵的操作都要用转换程序来进行转换。常用指针概念

及其数据类型如表 11.1 所示。

表 11.1 C 和 FORTRAN 程序中使用的指针类型

MatLab 指针	说 明	数据类型
ep	指向 MatLab 的过程	Engine
fp	文件指针	MATFile
mp	矩阵指针	Matrix
pr,pi	矩阵的实部和虚部	Double
ir,jc	矩阵的行和列的下标	Int

在 C 和 FORTRAN 中要用到许多转换函数，用来对矩阵、MatLab 工作空间和 MAT 文件进行操作。在 C 和 FORTRAN 中，这些函数基本相同，因而只列出一次，如有特殊情况，会注明。在支持 MatLab 和 C 及 FORTRAN 进行对话的库程序中，共有 4 类命令，这 4 类命令分别为：

mx　　矩阵命令
mat　　MAT 文件命令
eng　　MatLab 过程命令
mex　　MEX 命令

一般命令的第一组字母代表函数命令的类型。

11.1.1 MAT 文件命令

MAT 文件命令是在 C 和 FORTRAN 中用来读写 MatLab 的二进制文件的命令。在 MatLab 中，存储数据的格式按 MAT 文件的格式存储。MAT 文件可以存储一个或多个矩阵数据。数据在文件中以顺序格式存储，每个矩阵数据的开始是固定长度的矩阵信息，此信息记录了矩阵的特征信息，如类型、维数等，然后是矩阵的数据，这部分数据所占用的磁盘空间的大小由信息头中反映矩阵大小的信息决定与 MAT 文件操作有关的函数列表如表 11.2 所示。

表 11.2 与 MAT 文件操作有关的函数列表

函数指令	含 义
MATFile *matOpen(const char *filename, const char *mode)	函数打开一个名为 filename 的 MAT 文件，filename 为字符串类型，函数返回一个指向文件的指针 fp，字符串 mode 代表打开文件的模式，可以取下列值：'r'代表“读文件”；'w'代表“写文件”；'u'代表“更新文件”
int matClose(MATFile *fp)	关闭指针 fp 所指向的 MAT 文件
mxArray *matGetArray(MATFile *fp, const char *name)	从 MAT 文件 fp 中读取名为 namestr 的变量，并返回一个指向矩阵的指针 mp
matGetMatrix (fp)	这是 MatLab 4 版本中的命令

续表

函数指令	含 义
int matPutArray(MATFile *fp, const mxArray *mp)	向 MAT 文件 fp 中存入矩阵 mp
MatPutMatrix(fp)	这是 MatLab 4 版本中的命令
mxArray *matGetNextArray(MATFile *fp)	从 MAT 文件 fp 中读入下一个矩阵并返回指针 mp
int matDeleteArray(MATFile *fp, const char *name)	从文件 fp 中删除名为 namestr 的变量
MatGetFull(fp,namestr,m,n,pr,pi)	从 MAT 文件 fp 中读取名为 namestr 的 m×n 矩阵，并将 pr 指向矩阵的实部，将 pi 指向矩阵的虚部(这是 MatLab 4 版本中的命令)
MatPutFull(fp,anmestr,m,n,pr,pi,)	向文件 fp 中写入名为 namestr 的矩阵，指针 pr 和 pi 分别指向待存矩阵的实部和虚部(这是 MatLab4 版本中的命令)
MatGetString(fp,namestr,str,strlen)	从文件 fp 中读取名为 namestr 的变量到字符串变量 str 中(这是 MatLab 4 版本中的命令)
MatPutString(fp,namestr,str)	将字符串变量 str 写入 MAT 文件中，变量名为 namestr(这是 MatLab 4 版本中的命令)
mxArray *matGetArrayHeader(MATFile *fp, const char *name)	从 MAT 文件 fp 中获得名为 name 的矩阵的头信息，即除 pr，pi，ir，jc 之外的所有信息
char **matGetDir(MATFile *fp, int *num)	此函数用来获得一个 MAT 文件中的所有矩阵的名的列表，Num 为包含此 MAT 文件中的矩阵个数的变量的地址
FILE *matGetFp(MATFile *fp)	此函数用来得到 C 文件的句柄
mxArray *matGetNextArrayHeader(MATFile *fp)	获得下一个矩阵的信息头

11.1.2 mx 矩阵操作命令

在 MatLab 中只有一种数据类型，就是矩阵，MatLab 中矩阵的格式与 C 或 FORTRAN 中的数组的存放格式不太一样，因而 MatLab 中的矩阵不能直接用在 C 或 FORTRAN 语言中，而 C 或 FORTRAN 语言中一般格式的数组也不能直接被 MatLab 使用。它们之间交换数据必须存在一种机制。Mx 矩阵操作命令就是用来实现这一机制的，它使得 MatLab 与 C 或 FORTRAN 之间进行数据交换成为可能。此部分的函数指令如表 11.3 所示。

表 11.3 矩阵操作函数列表

函数指令	含 义
int mxGetM(const mxArray * mp)	得到矩阵 mp 的行数
int mxGetN(const mxArray * mp)	得到矩阵 mp 的列数
double *mxGetPr(const mxArray * mp)	返回指向矩阵 mp 的实部的指针 pr
double *mxGetPi(const mxArray * mp)	返回指向矩阵 mp 的虚部的指针 pi

续表

函数指令	含　义
void mxSetPr(mxArray * mp, double *pr)	将矩阵 mp 的实部返回给指针 pr(此函数为 FORTRAN 中专用)
void mxSetPi(mxArray * mp, double *pi)	将矩阵 mp 的虚部返回给指针 pi(此函数为 FORTRAN 中专用)
const char *mxGetName(const mxArray * mp)	返回一个指向矩阵 mp 的变量名的串指针
void mxSetName(mxArray * mp, const char *name)	将 mp 的变量名指针赋值给串指针 name(此函数为 FORTRAN 中专用)
int mxGetNzmax(const mxArray * mp)	得到稀疏矩阵 mp 的非零元素的最大值
void mxSetNzmax(mxArray * mp, int nzmax)	将稀疏矩阵 mp 的非零元素的最大值赋值给 nzmax(此函数为 FORTRAN 中专用)
int *mxGetIr(const mxArray * mp)	返回指针 ir，指向稀疏矩阵 mp 的行下标向量
int *mxGetJc(const mxArray * mp)	返回指针 jc，指向稀疏矩阵 mp 的列下标向量
void mxSetIr(mxArray * mp, int *ir)	将指针 ir 指向稀疏矩阵 mp 的行下标(此函数为 FORTRAN 中专用)
void mxSetJc(mxArray * mp, int *jc)	将指针 jc 指向稀疏矩阵 mp 的列下标(此函数为 FORTRAN 中专用)
MxIsFull(mp)	如果矩阵 mp 为全矩阵，函数返回 1，否则返回 0(这是 MatLab 4 版本中的命令)
bool mxIsSparse(const mxArray * mp)	如果矩阵 mp 为稀疏矩阵，函数返回 1，否则返回 0
MxIsString(mp)	如果矩阵 mp 为字符串矩阵，函数返回 1，否则返回 0(这是 MatLab 4 版本中的命令)
bool mxIsNumeric(const mxArray * mp)	如果矩阵 mp 为数矩阵，函数返回 1，否则返回 0
MxIsTypeDouble(mp)	如果矩阵 mp 为双精度存储的，则函数返回 1，否则返回 0 在现在的机器上，矩阵都是以双精度存储的，因而函数恒返回 1
bool mxIsComplex(const mxArray * mp)	如果矩阵 mp 为复数矩阵，函数返回 1，否则返回 0
MxCreatefull(m,n,cf)	函数返回一个全 m×n 阵的指针，如果 cf 为 0，则矩阵为实矩阵，否则为复数矩阵 (这是 MatLab 4 版本中的命令)
mxArray *mxCreateSparse(int m, int n, int nzmax, mxComplexity ComplexFlag)	函数返回一个 m×n 稀疏阵的指针，其中有 nzmax 个非零元素，如果 ComplexFlag 为零则矩阵为实矩阵，否则为复数矩阵
MxFreeMatrix(mp)	释放矩阵 mp 所占有的内存(此函数为 FORTRAN 中专用，这是 MatLab 4 版本中的命令)

续表

函数指令	含义
void *mxCalloc(size_t n, size_t size)	分配一块内存空间，并返回一个指向此空间的指针，Size 是一个元素占用的字节数，n 是元素的个数，则分配的内存空间大小为 size×nm 字节
void mxFree(void *ptr)	返回指针 ptr 所指向的内存空间的大小，必须是用函数 mxCalloc()分配的空间(此函数为 FORTRAN 专用)
int mxGetString(const mxArray * mp, char *buf, int buflen)	从 mp 中读入长度小于或等于 strlen 的字符串，并赋值给 str,. 变量 str 必须事先被赋给存储空间在 FORTRAN 中，str 就是正常的字符串变量
mxArray *mxCreateString(const char *str)	创建一个包含 str 内容的矩阵 mp
double mxGetScalar(const mxArray * mp)	返回矩阵 mp 的元素(1,1)的值

11.1.3 引擎操作指令

引擎操作指令主要是 C 或 FORTRAN 程序对 MatLab 进行控制的一些指令，如从 MatLab 的工作空间中读取或存入数据、控制 MatLab 执行相应的命令等。本质上说就是 C 或 FORTRAN 语言控制 MatLab 的控制指令。此部分的函数指令如表 11.4 所示。

表 11.4 引擎操作函数列表

函数指令	含义
integer*4 function engOpen(startcmd)	开始一个 MatLab 引擎 ep，字符串 startcmd 包含了开始 MatLab 的指令，如果有必要，也要包含路径
integer*4 function engClose(ep)	关闭引擎 ep，如果引擎正常关闭，函数返回 0，否则返回 1
integer*4 function engGetMatrix(ep, name)	从 MatLab 引擎的工作空间中读入一个名为 name 的矩阵，返回给矩阵变量 mp
integer*4 function engPutMatrix(ep, mp)	在 MatLab 的工作空间中存入矩阵 mp
integer*4 function engGetFull(ep, name, m, n, pr, pi)	从引擎 ep 的工作空间中读入一个 m×n 的全矩阵，但不返回给矩阵指针 mp，而是将矩阵的实部和虚部分别赋值给指针 pr 和 pi
integer*4 function engPutFull(ep, name, m, n, pr, pi)	将以指针 pr 和 pi 所指向的矩阵所组成的 m×n 的名为 name 的全矩阵存入引擎 ep 的工作空间中
integer*4 function engEvalString(ep, command)	在 MatLab 引擎 ep 的工作空间中执行由字符串 command 所指定的 MatLab 命令
subroutine engOutputBuffer(ep, p, n)	分配一块用指针 p 指向的缓冲区，此缓冲区是被引擎 ep 用作输出用，其最大字节数是 n

11.1.4 MEX 指令

mex 指令是用来构造 MEX 函数的一些指令。它是在 MatLab 运行时，被调用的 MEX 函数与 MatLab 之间进行数据交换和控制转移的一些指令，如从 MatLab 读取或存入数据、执行 MatLab 的函数、向 MatLab 输出出错信息等。此外 mexFunction 函数是定义 MEX 函数时所必需的接口函数，其他相关函数指令如表 11.5 所示。

表 11.5 构造 MEX 函数所用到的函数列表

函数指令	含 义
void mexFunction(int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])	此函数在每个 MEX 文件中都存在，MatLab 在调用 MEX 函数时，先调用此函数，然后再由此函数来调用所需要的函数，此函数主要是对输入输出的参数进行处理，其 4 个参数的意义分别为：nls 和 nrs 是输入输出参数的个数；prhs 和 plhs 是分别指向输入输出参数的指针
int mexCallMatLab(int nlhs, mxArray *plhs[], int nrhs, mxArray *prhs[], const char *command_name)	调用 MatLab 中名为 command_name 的指令、M 函数和其他的 MEX 函数，其中 nlhs 为输出变量的个数，nrhs 为输入变量的个数，Plhs[]为指向输出变量的指针，prhs[]为指向输入变量的指针
int mexEvalString(const char *command)	执行 MatLab 中的命令 command
const mxArray *mexGetArrayPtr(const char *name, const char *workspace)	返回 MatLab 工作空间中的名为 namestr 的矩阵，返回的是一个矩阵指针 mp
void mexErrMsgTxt(const char *error_msg)	向 MatLab 输出内容为 errstr 的错误信息，并将控制权交给 MatLab
mxArray *mexGetArray(const char *name, const char *workspace)	从 MatLab 的工作空间取名为 name 的矩阵，函数返回矩阵指针 mp。Word 可以为"base"从当前工作空间中查找；"caller"从调用此函数的工作空间中查找；"global"从全局变量中查找
int mexPutArray(mxArray *array_ptr, const char *workspace)	向工作空间中保存矩阵 mp
MexGetFull(namestr,m,n,pr,pi)	从工作空间中取得名为 namestr 的 m×n 的全矩阵，分别将其实部和虚部返回给指针 pr 和 pi(这是 MatLab 4 版本中的命令)
MexPutFull(namestr,m,n,pr,pi)	将由 pr 和 pi 所定义的 m×n 的全矩阵以变量名 namestr 的形式存入工作空间(这是 MatLab 4 版本中的命令)
int mexPrintf(const char *format, str1, ---, strn)	以格式字符串 format 所指定的格式输出字符串 str1,…, strn，其如同在工作空间中用 sprintf()命令输出字符串

11.1.5 接口程序的结构

现在先讨论 MatLab 与其他语言接口的问题，MatLab 与 C 和 FORTRAN 的接口程序的结构和实现过程都一样，在此一并作介绍。

MatLab 在处理与其接口的外部程序时，所使用的方法都是一样的，如在 M 文件中或在工作空间的命令窗口中调用 fun 函数：[x,y,z]=fun(a,b,c,d)，在此设 fun 是由 C 或 FORTRAN 编写的 MEX 函数，MatLab 会先检索其内置函数、用户的 M 文件和用户的 MEX 文件，如果发现 fun 是一个 MEX 文件，则可进行下面的操作：

(1) 将函数的输入变量的个数赋值给变量 nrhs，在此例中 nrhs 值应该为 4。

(2) 将函数返回变量的数目赋值给 nlhs，此例中 nlhs 值为 3。

(3) 生成 2 个整数矩阵，用来分别存放输入输出参数的地址，这 2 个整数矩阵是指向输入输出参数的指针数组。输入参数的指针是 prhs，返回参数的指针是 plhs。在此例中，prhs(1)、prhs(2)、prhs(3)、prhs(4)分别包含变量 a,b,c,d 的地址；plhs(1)、plhs(2)、plhs(3)分别包含输出参数 x，y，z 的地址。

在此，需要说明以下几点：

- 在调用函数时，被调用的函数并不能改变输入参数的值。如函数 fun 并不能改变参数 a，b，c，d 的值，实际上，MatLab 在调用函数时先将这些变量拷贝到被调用函数的临时内存空间中。
- 在调用函数时，输入的参数个数可以少于定义值，如可以这样调用函数 fun(a,b)，fun(a)或 fun()，当然也要在函数内部指定未赋值参数的默认值。
- 在 MatLab 中，本质上说只有一种数据结构，就是矩阵，所有的物理或对象都是以矩阵的形式存放的，标量和向量都是矩阵的特殊形式。矩阵本身有许多特性，如矩阵具有实或复的特性，有行数和列数等。指向 MatLab 的矩阵的指针实际上是指向存放矩阵元素本身的值及反映矩阵特性的数据的一块连续内存的首地址，MatLab 中的矩阵的概念可以看成是 C 语言中的一种结构体。
- MatLab 的矩阵数据是按列存放的，这与 FORTRAN 相同，但与 C 不同。

MatLab 的标准发布版中都含有一些服务性的程序库，用以处理与 FORTRAN 或 C 的对话，这些程序大都处理一些与数据结构有关的操作。在 MEX 文件中含有一个接口程序用来充当 MatLab 与 C 或 FORTRAN 对话的桥梁，一般名为 mexFunction，主要完成以下几个任务：分配用以过渡或交换数据的矩阵空间；检查输入输出参数的值的合法性；给出输入参数的默认值；捕捉并输出错误信息。

MatLab 的 MEX 文件的扩展名可以为 MEX 和 DLL，是由 C 或 FORTRAN 源程序经过编译生成的动态链接子程序，其调用过程类似于 MatLab 的内置函数。使用 DLL 文件可以直接访问 Windows 资源的各种功能，可以生成基于 Windows 的用户图形界面，并利用 Windows 的 DDE 功能与其他应用程序进行数据交换。在 Windows 95/98 下，MEX 文件是 32 位的 DLL 格式，因而在编译时最好用 32 位的编译器和链接器来生成。MatLab 支持的 32 位编译器有 Microsoft C/C++ Compiler (ver:7.0)、Borland C++ Ver 4.5 及 NDP FORTRAN Ver 3.0。

MatLab 调用函数的顺序是先执行 MEX 文件，其次是 DLL 文件，最后是 M 文件。为

了方便快捷地编译生成 MEX 文件，推荐使用 mex -setup 让 MatLab 自动设置有关编译器等方面的参数以及编译过程，然后在 MatLab 命令窗口输入 mex filename，或在 DOS 提示符下输入 Cmex filename 来直接编译生成 MEX 文件。在 MatLab 目录下的 BIN 子目录下，有一个 CMEX.BAT 文件，它是建立 C 和 MEX 文件的批处理文件。

11.2 MatLab 与 FORTRAN 语言的接口

在 C 中，一般参数的传递是按值传递的(用指针传递参数除外)，即在子程序中修改参数的值并不影响主程序中的参数，而 FORTRAN 中的参数传递是按地址传递的，在子程序中也可以修改从主程序中传递下来的参数的值。如果要在 FROTRAN 中使用按值传递参数，则应该使用%val 构造，它是对 FORTRAN 的扩展。如果所使用的 FORTRAN 编译器支持%val 构造，就可以直接使用数据指针将指针的内容直接传递到子程序的类型为双精度矩阵的参数变量中；如果 FORTRAN 不支持%val 结构，则需用 mxCopy 类函数访问指针的内容。例如，对于函数 fsum，如果编译器支持%val 构造，则可以这样访问函数：call fsum(%val(A)，%val(B)，%val(C)，%val(D))。但是如果编译器不支持%val 构造，则程序的调用过程就复杂一些，如上面的参数都是 5×5 的矩阵，则在主程序中首先声明 4 个当地矩阵 real*8，a(5,5)，b(5,5)，c(5,5)，d(5,5)，后将传递的指针变量的值拷贝到当地变量：call mxCopyPtrToReal8(A,a,25)

```
call mxCopyPtrToReal8(B,b,25)
```

然后调用子程序：call fsum(a,b,c,d)，再将计算结果传递给输出变量：

```
call mxCopyReal8ToPtr(c,C,25)
call mxCopyReal8ToPtr(c,C,25)
```

11.2.1 从 FORTRAN 中调用 MatLab 命令

要在 FORTRAN 程序中使用 MatLab 的命令，先要用 engOpen 打开一个引擎，然后将要在 MatLab 中使用的矩阵存入 MatLab 的工作空间中。用 mxGreateFull 或 mxGreateSpare 创建 MatLab 格式的矩阵，用 mxSetName 给矩阵命名，然后将 FORTRAN 格式的矩阵拷贝给 MatLab 格式的矩阵。建立和处理好矩阵后，用 engPutMatrix 将矩阵存入工作空间中，这时 MatLab 便可以接收命令了，可以像直接操作 MatLab 的工作空间那样执行命令，只是此时的命令要由函数 engEvalString 来传递。这样，MatLab 与 FORTRAN 语言直接的连续通道就建立了。

库函数 libmat.a 中含有所有的支持 C 和 MatLab 对话的程序，编译命令中，库函数的路径和编译命令-lnsl 与机器的不同有关，可以通过阅读相关的帮助文件来确定，其余的文件路径也是如此。

为了便于理解和编译程序，给出 MatLab 自带的一些例子，并给予详细的说明：

```
C         文件名：fengdemo.f
C          这个简单的例子用来说明怎样从 FORTRAN 程序中调用 MatLab 的引擎
```

```
C
C          Copyright (c) 1996-1998 by The MathWorks, Inc.
C          All Rights Reserved.
C=================================================================
C $Revision: 1.3 $
      program main
C-----------------------------------------------------------------
C     ( integer ) Replace integer by integer*8 on the DEC Alpha and the
C     SGI 64 platforms
C
       integer engOpen, engGetMatrix, mxCreateFull, mxGetPr
       integer ep, T, D, result
C-----------------------------------------------------------------
C
C        下面是变量的声明
      double precision time(10), dist(10)
      integer stat, temp
      data time / 1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0 /

C     打开一个引擎
      ep = engOpen('MatLab ')
C     如果返回结构不为 0，则说明打开引擎出错
      if (ep .eq. 0) then
        write(6,*) 'Can''t start MatLab engine'
        stop
      endif
C     生成一个实矩阵，并赋名为'T'，将矩阵 time 的值拷贝给 T
      T = mxCreateFull(1, 10, 0)
      call mxSetName(T, 'T')
      call mxCopyReal8ToPtr(time, mxGetPr(T), 10)
C
C     将矩阵 T 存入 MatLab 的工作空间，便于以后调用和处理
C
      call engPutMatrix(ep, T)
C
C     执行函数 distance = (1/2)g.*t.^2
C     (g 是重力加速度)
C
      call engEvalString(ep, 'D = .5.*(-9.8).*T.^2;')
C
C     将结果用图形再现，用 engEvalString 在 MatLab 工作空间中执行绘图命令
      call engEvalString(ep, 'plot(T,D);')
      call engEvalString(ep, 'title(''Position vs. Time'')')
      call engEvalString(ep, 'xlabel(''Time (seconds)'')')
      call engEvalString(ep, 'ylabel(''Position (meters)'')')
C
C     read from console to make sure that we pause long enough to be
```

```
C     able to see the plot
C
      print *, 'Type 0 <return> to Exit'
      print *, 'Type 1 <return> to continue'
      read(*,*) temp
      if (temp.eq.0) then
         print *, 'EXIT!'
         stop
      end if
C
      call engEvalString(ep, 'close;')
C     从当前的工作空间中获取矩阵 D，并将其赋值给 dist
      D = engGetMatrix(ep, 'D')
      call mxCopyPtrToReal8(mxGetPr(D), dist, 10)
C     打印输出矩阵 time 和 dist
      print *, 'MatLab computed the following distances:'
      print *, '  time(s)  distance(m)'
      do 10 i=1,10
         print 20, time(i), dist(i)
 20      format(' ', G10.3, G10.3)
 10   continue
C     释放内存空间
      call mxFreeMatrix(T)
      call mxFreeMatrix(result)
      stat = engClose(ep)
C
      stop
      end
```

11.2.2 用 FORTRAN 读写 MAT 文件

在 FORTRAN 程序中，要从 MatLab 生成的二进制文件 MAT 文件中读写数据，则必须有一个通道使得 FORTRAN 和 MAT 文件中的矩阵相互转换，要通过一系列的函数来实现。在对 MAT 文件进行读写操作时，先要打开一个文件，在对文件操作完毕后要关闭文件。

```
C     matdemo1.f
C     文件名
C     This is a simple program that illustrates how to call the MatLab
C     MAT-file functions from a FORTRAN program.  This demonstration
C     focuses on writing MAT-files.
C
C Copyright (c) 1996-1998 by The MathWorks, Inc.
C All Rights Reserved.
C=====================================================================
C $Revision: 1.3 $

C     matdemo1 - create a new MAT-file from scratch.
```

```
C
      program matdemo1
C-----------------------------------------------------------------
C     ( integer ) Replace integer by integer*8 on the DEC alpha and the
C     SGI64 platforms
C       下面是函数和变量的声明
      integer matOpen, mxCreateFull, mxCreateString
      integer matGetMatrix, mxGetPr
      integer mp, pa1, pa2, pa3
C-----------------------------------------------------------------
C     Other variable declarations here
C
      integer status, matClose
      double precision dat(9)
      data dat / 1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0 /
C
C     打开一个名为matdemo.mat 的文件用以写入数据
C
      write(6,*) 'Creating MAT-file matdemo.mat ...'
      mp = matOpen('matdemo.mat', 'w')
C      如果返回结果不为 0，则说明打开文件出错
      if (mp .eq. 0) then
         write(6,*) 'Can''t open ''matdemo.mat'' for writing.'
         write(6,*) '(Do you have write permission in this directory?)'
         stop
      end if
C
C     生成新的 3×3 的实矩阵变量，为其分配内存空间，并命名为 Numeric
C
      pa1 = mxCreateFull(3,3,0)
      call mxSetName(pa1, 'Numeric')
C      生成字符串变量，命名为 String
      pa2 = mxCreateString('MatLab: The language of computing')
      call mxSetName(pa2, 'String')
C      生成字符串变量，命名为 String2
      pa3 = mxCreateString('MatLab: The language of computing')
      call mxSetName(pa3, 'String2')
C      将新生成的 3 个变量存入 MAT 文件中
      call matPutMatrix(mp, pa1)
      call matPutMatrix(mp, pa2)
      call matPutMatrix(mp, pa3)
C
C     将矩阵 dat 的值拷贝给矩阵 pa1，然后重新向文件写入矩阵 pa1，这一操作将覆盖
C     上面刚写入的矩阵 pa1
      call mxCopyReal8ToPtr(dat, mxGetPr(pa1), 9)
      call matPutMatrix(mp, pa1)
C
```

```
C     从 MAT 文件中删除变量 String2
C
      call matDeleteMatrix(mp, 'String2')
C
C     关闭 MAT 文件,如果返回结果不为 0, 则说明文件没有正常关闭
C
      status = matClose(mp)
      if (status .ne. 0) then
         write(6,*) 'Error closing MAT-file'
         stop
      end if
C     以只读的方式重新打开 matdemo.mat 文件
      mp = matOpen('matdemo.mat', 'r')
      if (status .ne. 0) then
         write(6,*) 'Can''t open ''matdemo.mat'' for reading.'
         stop
      end if
C     从文件中读入上面存入的名为 Numeric 的变量,并判断矩阵是否为数据型
      pa1 = matGetMatrix(mp, 'Numeric')
      if (mxIsNumeric(pa1) .eq. 0) then
         write(6,*) 'Invalid non-numeric matrix written to MAT-file'
         stop
      end if
C     从文件中读入名为 String 的矩阵变量, 并判断是否为字符型矩阵
      pa2 = matGetMatrix(mp, 'String')
      if (mxIsString(pa2) .eq. 0) then
         write(6,*) 'Invalid non-numeric matrix written to MAT-file'
         stop
      end if
C     从文件中读入名为 String2 的矩阵变量, 如果返回结果不为零说明变量 String2 存
在
      pa3 = matGetMatrix(mp, 'String2')
      if (pa3 .ne. 0) then
         write(6,*) 'String2 not deleted from MAT-file'
         stop
      end if
C
C     释放 3 个变量所占用的内存
      call mxFreeMatrix(pa1)
      call mxFreeMatrix(pa2)
      call mxfreeMatrix(pa3)
C     关闭文件, 并判断是否正常关闭
      status = matClose(mp)
      if (status .ne. 0) then
         write(6,*) 'Error closing MAT-file'
         stop
      end if
```

```
C
      write(6,*) 'Done creating MAT-file'
      stop
      end
```

11.2.3 在 MatLab 中使用由 FORTRAN 编写的 MEX 函数

在 MatLab 中，调用 MEX 函数的形式与调用 M 函数的形式完全相同，所不同的是函数内部的实现过程也要进行一系列的数据格式的转换，并且要对输入输出参数进行管理。在 MEX 文件中，矩阵类型和关于矩阵操作的命令与前面介绍的大都相同。在 FORTRAN 的源程序中要有两个独立的函数，一个名为 mexFunction，这是 MatLab 所调用的主函数，其主要作用是实现输入输出参数的管理，以及调用其他的函数。此函数有 4 个参数，其中 nrhs 和 nlhs 分别表示输入输出参数的个数，prhs 和 plhs 为分别指向输入输出参数的指针数组。

C 此函数是 MatLab 与 FORTRAN 的接口函数

```
C YPRIMEG.FOR - Gateway function for YPRIME.FOR
C
C This is an example of the FORTRAN code required for interfacing
C a .MEX file to MatLab.
C
C This subroutine is the main gateway to MatLab.  When a MEX function
C  is executed MatLab calls the MEXFUNCTION subroutine in the corresponding
C  MEX file.
C
C Copyright (c) 1984-1998 by The MathWorks, Inc.
C All Rights Reserved.
C $Revision: 1.5 $
C
      SUBROUTINE MEXFUNCTION(NLHS, PLHS, NRHS, PRHS)

C-----------------------------------------------------------------------
C     (integer) Replace integer by integer*8 on the DEC Alpha and the
C     SGI 64-bit platforms
C      声明输入参数 PLHS 和 PRHS
      INTEGER PLHS(*), PRHS(*)
C-----------------------------------------------------------------------
C      声明输入参数 NLHS 和 NRHS
      INTEGER NLHS, NRHS
C
C-----------------------------------------------------------------------
C     (integer) Replace integer by integer*8 on the DEC Alpha and the
C     SGI 64-bit platforms
C      声明用到的子例程子程序
      INTEGER MXCREATEFULL, MXGETPR
      INTEGER MXGETM, MXGETN
```

```
C
C 上面的变量、参数和子例程子程序的声明在所有的 MEX 文件中都是必需的
C------------------------------------------------------------------
C
C------------------------------------------------------------------
C     (integer) Replace integer by integer*8 on the DEC Alpha and the
C     SGI 64-bit platforms
C     下面是变量声明
      INTEGER YPP, TP, YP
      INTEGER M, N
      REAL*8 RYPP(4), RTP, RYP(4)

C
C 对输入及输出参数进行校验，判断其合法性
C
      IF (NRHS .NE. 2) THEN
        CALL MEXERRMSGTXT('YPRIME requires two input arguments')
      ELSEIF (NLHS .GT. 1) THEN
        CALL MEXERRMSGTXT('YPRIME requires one output argument')
      ENDIF
C
C 检查第 2 个输入参数的维数，应该为 1×4 或 4×1 的矩阵
C
      M = MXGETM(PRHS(2))
      N = MXGETN(PRHS(2))
C
      IF ((MAX(M,N) .NE. 4) .OR. (MIN(M,N) .NE. 1)) THEN
        CALL MEXERRMSGTXT('YPRIME requires that Y be a 4 x 1 vector')
      ENDIF
C
C 为返回参数分配内存空间
C
      PLHS(1) = MXCREATEFULL(M,N,0)
C
C 为输出和输入参数分配指针
C
      YPP = MXGETPR(PLHS(1))
      TP = MXGETPR(PRHS(1))
      YP = MXGETPR(PRHS(2))
C
C 将输入参数的值拷贝给当地变量
      CALL MXCOPYPTRTOREAL8(TP, RTP, 1)
      CALL MXCOPYPTRTOREAL8(YP, RYP, 4)
C
C 调用子程序对数据进行计算，并得到计算结果保存在当地变量 RYPP 中
C
      CALL YPRIME(RYPP,RTP,RYP)
```

```
C
C 将当地变量 RUPP 的值拷贝给返回参数 YPP
     CALL MXCOPYREAL8TOPTR(RYPP, YPP, 4)
C 至此程序执行完毕
     RETURN
     END

C 此函数是被接口程序调用的用来进行计算的子程序
C The actual YPRIME subroutine in FORTRAN
C Copyright (c) 1984-1998 by The MathWorks, Inc.
C All Rights Reserved.
C $Revision: 1.2 $
C 程序及参数和变量的声明
     SUBROUTINE YPRIME(YP, T, Y)
     REAL*8 YP(4), T, Y(4)
     REAL*8 MU, MUS, R1, R2
C 计算过程
     MU = 1.0/82.45
     MUS = 1.0 - MU
     R1 = SQRT((Y(1)+MU)**2 + Y(3)**2)
     R2 = SQRT((Y(1)-MUS)**2 + Y(3)**2)
     YP(1) = Y(2)
     YP(2) =2*Y(4) +Y(1) -MUS*(Y(1)+MU)/(R1**3) -MU*(Y(1)-MUS)/(R2**3)
     YP(3) = Y(4)
     YP(4) = -2*Y(2) + Y(3) - MUS*Y(3)/(R1**3) - MU*Y(3)/(R2**3)
C 程序执行结束
     RETURN
     END
```

11.3 MatLab 与 C 语言的接口

11.3.1 从 C 程序中使用 MatLab

从 C 程序中调用 MatLab，必须先用 engOpen()打开一个引擎，然后将要在 MatLab 中使用的矩阵存入 MatLab 的工作空间中。最好开始就以 MatLab 的格式处理矩阵，用 mxGreateFull 或 mxGreateSpare 创建矩阵，用 mxSetName 给矩阵命名。当然也可以以随意格式创建矩阵，然后将其转换为 MatLab 格式。C 语言与 MatLab 存储矩阵的方式是不同的，C 以行存储矩阵，而 MatLab 中，是按列存储的。建立和处理好矩阵后，用 engPutMatrix 将矩阵存入工作空间中，这时，MatLab 便可以接收命令了，可以像直接操作 MatLab 的工作空间那样执行命令，只是此时的命令要由函数 engEvalString 来传递。这样，MatLab 与 C 语言直接的连续通道就建立了。

库函数 libmat.a 中含有所有支持 C 和 MatLab 对话的程序，编译命令中，库函数的路径和编译命令-lnsl 与机器的不同有关，可以通过阅读相关的帮助文件来确定。

```
/* $Revision: 1.4 $ */
/* 文件名
 *  engdemo.c
 *  此函数用来说明怎样从 C 程序中调用 MatLab 的指令
 * Copyright (c) 1996-1998 by The MathWorks, Inc.
 * All rights reserved
 */
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include "engine.h"
#define  BUFSIZE 256
int main()
{
    Engine *ep;
    mxArray *T = NULL, *result = NULL;
    char buffer[BUFSIZE];
    double time[10] = { 0.0, 1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0 };
    /*
     * 用字符串"MatLab"打开当地的 MatLab 引擎
     * 如果要从远程主机上打开一个 MatLab 引擎，则应该使用远程主机的主机名作为传递
     * 参数，对于更复杂的情况，可以传递含有空格的字符串，MatLab 会逐一处理每组字
     * 符串，如果返回的结果为空，则说明操作不成功
     */
    if (!(ep = engOpen("\0"))) {
        fprintf(stderr, "\nCan't start MatLab engine\n");
        return EXIT_FAILURE;
    }
    /*
     * 第 1 部分，把数据存入 MatLab 的工作空间中，并对数据进行处理并绘图
     */
    /*
  * 生成新的 1×10 的实矩阵，并分配内存空间，赋名为 T，然后将矩阵 time 的值拷贝给
*矩阵 T
     */
    T = mxCreateDoubleMatrix(1, 10, mxREAL);
    mxSetName(T, "T");
    memcpy((void *)mxGetPr(T), (void *)time, sizeof(time));
    /*
     * 将矩阵 T 存入 MatLab 的工作空间中，等候处理
     */
    engPutArray(ep, T);
    /*
     * 执行函数 distance = (1/2)g.*t.^2 对数据进行处理
     * (g 是重力加速度)
     */
    engEvalString(ep, "D = .5.*(-9.8).*T.^2;");
```

```
        /*
         * 将计算结果以图形形式表示出来
         */
        engEvalString(ep, "plot(T,D);");
        engEvalString(ep, "title('Position vs. Time for a falling
object');");
        engEvalString(ep, "xlabel('Time (seconds)');");
        engEvalString(ep, "ylabel('Position (meters)');");
        /*
         * 让图形消失前停留足够长的时间
         */
        printf("Hit return to continue\n\n");
        fgetc(stdin);
        /*
         * 释放矩阵 T 所占用的内存空间，并关闭 MatLab 引擎
         */
        printf("Done for Part I.\n");
        mxDestroyArray(T);
        engEvalString(ep, "close;");
        /*
         * 第 2 部分
         *在此部分中，读入一个字符串命令，并执行此字符串命令，此命令将生成一个变量 X，
         * 然后分析这个变量，并判断其类型。这一过程将一直进行下去，直到用户输入的命
         * 令不能生成变量 X，或变量 X 被删除
         */
        /*
         * 用函数 engOutputBuffer 来捕获 MatLab 的输出数据，以便将其回显
         */
        engOutputBuffer(ep, buffer, BUFSIZE);
        while (result == NULL) {
            char str[BUFSIZE];
            /*
             * 从标准输入读入一个字符串命令
             */
            printf("Enter a MatLab command to evaluate.  This command
should\n");
            printf("create a variable X.  This program will then
determine\n");
            printf("what kind of variable you created.\n");
            printf("For example: X = 1:5\n");
            printf(">> ");
            fgets(str, BUFSIZE-1, stdin);
            /*
             * 执行此字符串命令
             */
            engEvalString(ep, str);
            /*
```

```
        * 显示命令窗口中的输出，前 2 个字符是提示符(>>)
        */
       printf("%s", buffer+2);
       /*
        * 获得用户生成的变量 X，并用函数 mxGetClassName()来判断变量的类型
        */
       printf("\nRetrieving X...\n");
       if ((result = engGetArray(ep,"X")) == NULL)
         printf("Oops! You didn't create a variable X.\n\n");
       else {
        printf("X is class %s\t\n", mxGetClassName(result));
       }
    }
    /*
     * 到此，程序结束，要释放矩阵占用的内存并关闭引擎
     */
    printf("Done!\n");
    mxDestroyArray(result);
    engClose(ep);
    return EXIT_SUCCESS;
}
```

11.3.2 用 C 语言读写 MAT 文件

在 C 程序中，要从 MatLab 生成的二进制文件 MAT 文件中读写数据，则必须有一个通道使得 C 和 MAT 文件中的矩阵相互转换，这也要通过一系列的函数来实现。在对 MAT 文件进行读写操作时，先要打开一个文件，对文件操作完毕后要关闭文件。

下面的程序同前面介绍的 FORTRAN 程序功能相同，可参照前面的程序来理解此程序。

```
/* $Revision: 1.3 $ */
/*
 * MAT-file creation program
 * See the MatLab API Guide for compiling information.
 * Calling syntax:
 * 程序名
 * matcreat
 * 此函数用以创建一个 MAT 文件
 *
 * Copyright (c) 1996-1998 by The MathWorks, Inc.
 * All Rights Reserved.
 */

#include <stdio.h>
#include <stdlib.h>
#include "mat.h"
#define BUFSIZE 255
int create(const char *file) {
  MATFile *pmat;
```

```
mxArray *pa1, *pa2, *pa3;
double data[9] = { 1.0, 4.0, 7.0, 2.0, 5.0, 8.0, 3.0, 6.0, 9.0 };
char str[BUFSIZE];
printf("Creating file %s...\n\n", file);
pmat = matOpen(file, "w");
if (pmat == NULL) {
  printf("Error creating file %s\n", file);
  printf("(do you have write permission in this directory?)\n");
  return(1);
}
pa1 = mxCreateDoubleMatrix(3,3,mxREAL);
mxSetName(pa1, "LocalDouble");
pa2 = mxCreateDoubleMatrix(3,3,mxREAL);
mxSetName(pa2, "GlobalDouble");
memcpy((void *)(mxGetData(pa2)), (void *)data, 3*3*sizeof(double));
pa3 = mxCreateString("MatLab: the language of technical computing");
mxSetName(pa3, "LocalString");
matPutArray(pmat, pa1);
matPutArrayAsGlobal(pmat, pa2);
matPutArray(pmat, pa3);
/*
 * 现在修改矩阵 pa1 的值，并将其重新存入文件，这将覆盖前面存入的数据，
 * 也就是说，如果要存入的变量与文件中已经存在的变量重名，则新存入的变量会覆盖
 * 已经存在的变量
 */
memcpy((char *)(mxGetPr(pa1)), (char *)data, 3*3*sizeof(double));
matPutArray(pmat, pa1);
/* 释放矩阵变量所占用的内存空间 */
mxDestroyArray(pa1);
mxDestroyArray(pa2);
mxDestroyArray(pa3);
/*关闭文件*/
if (matClose(pmat) != 0) {
  printf("Error closing file %s\n",file);
  return(1);
}
/*
 * 重新打开文件
 */
pmat = matOpen(file, "r");
if (pmat == NULL) {
  printf("Error reopening file %s\n", file);
  return(1);
}
/*
 * 重新读出上面存入的变量,并验证上面的操作是否正确
 */
pa1 = matGetArray(pmat, "LocalDouble");
```

```
  if (pa1 == NULL) {
    printf("Error reading existing matrix LocalDouble\n");
    return(1);
  }
  if (mxGetNumberOfDimensions(pa1) != 2) {
    printf("Error saving matrix: result does not have two dimensions\n");
    return(1);
  }
  pa2 = matGetArray(pmat, "GlobalDouble");
  if (pa2 == NULL) {
    printf("Error reading existing matrix LocalDouble\n");
    return(1);
  }
  if (!(mxIsFromGlobalWS(pa2))) {
    printf("Error saving global matrix: result is not global\n");
    return(1);
  }
  pa3 = matGetArray(pmat, "LocalString");
  if (pa3 == NULL) {
    printf("Error reading existing matrix LocalDouble\n");
    return(1);
  }
  mxGetString(pa3, str, BUFSIZE);
  if (strcmp(str, "MatLab: the language of technical computing")) {
    printf("Error saving string: result has incorrect contents\n");
    return(1);
  }
  /* 清除变量所占用的内存空间 */
  mxDestroyArray(pa1);
  mxDestroyArray(pa2);
  mxDestroyArray(pa3);
 /*关闭文件*/
  if (matClose(pmat) != 0) {
    printf("Error closing file %s\n",file);
    return(1);
  }
  printf("Done\n");
  return(0);
}
/*下面是主函数*/
int main()
{
  int result;
  result = create("mattest.mat");
  return (result==0)?EXIT_SUCCESS:EXIT_FAILURE;
}
```

11.3.3 在 MatLab 中使用由 C 编写的 MEX 函数

在 C 的源程序中要有一个名为 mexFunction 的函数，这是 MatLab 所调用的主函数，其主要作用是对输入输出参数进行管理，以及调用其他函数。函数的定义结构为：

```
void mexFunction(int nlhs,mxArray *plhs[],int nrhs, const mxArray *prhs[])
```

{ /*程序代码*/}，对于如[x,y]=fsum(a,b,c)，在函数被调用之前，输出变量 x 和 y 未被创建，所以输出参数指针是执行空的，而输入参数 a,b,c 已经确定，输入参数的指针就分别指向这 3 个变量的地址。此函数有 4 个参数，其中 nrhs 和 nlhs 分别表示输入输出参数的个数，prhs 和 plhs 分别指向输入输出参数的指针数组。

```
/* $Revision: 1.2 $ */
/*
 * YPRIME.C 这是 YPRIME.M 函数的 MEX 文件，它的执行速度要比 YPRIME.M 要快
 * 其调用的语法为：
 *      [yp] = yprime(t, y)
 * Copyright (c) 1984-1998 by The MathWorks, Inc.
 * All Rights Reserved.
 */
#include <math.h>
#include "mex.h"
/* 定义输入参数 */
#define T_IN       prhs[0]
#define Y_IN    prhs[1]
/* 定义输出参数 */
#define YP_OUT  plhs[0]
/*定义宏命令*/
#if !defined(max)
#define max(A, B)   ((A) > (B) ? (A) : (B))
#endif
#if !defined(min)
#define min(A, B)   ((A) < (B) ? (A) : (B))
#endif
/*定义常数*/
#define pi 3.14159265
static  double  mu = 1/82.45;
static  double  mus = 1 - 1/82.45;
static void yprime(
           double   yp[],
           double   *t,
           double   y[]
           )
{
 double r1,r2;
 r1 = sqrt((y[0]+mu)*(y[0]+mu) + y[2]*y[2]);
 r2 = sqrt((y[0]-mus)*(y[0]-mus) + y[2]*y[2]);
```

```
        yp[0] = y[1];
        yp[1]                                                           =
    2*y[3]+y[0]-mus*(y[0]+mu)/(r1*r1*r1)-mu*(y[0]-mus)/(r2*r2*r2);
        yp[2] = y[3];
        yp[3] = -2*y[1] + y[2] - mus*y[2]/(r1*r1*r1) - mu*y[2]/(r2*r2*r2);
        return;
      }
      /*下面是 MEX 函数的接口函数*/
      void mexFunction(int nlhs,mxArray *plhs[],int nrhs, const mxArray
    *prhs[])
      {
        double*yp;
        double*t,*y;
        unsigned int  m,n;
        /* 检查参数个数的合法性 */
        if (nrhs != 2) {
          mexErrMsgTxt("YPRIME requires two input arguments.");
        } else if (nlhs > 1) {
          mexErrMsgTxt("YPRIME requires one output argument.");
        }
        /* 检查参数维数的合法性 */
        m = mxGetM(Y_IN);
        n = mxGetN(Y_IN);
        if (!mxIsNumeric(Y_IN) || mxIsComplex(Y_IN) ||
            mxIsSparse(Y_IN)  || !mxIsDouble(Y_IN) ||
            (max(m,n) != 4) || (min(m,n) != 1)) {
          mexErrMsgTxt("YPRIME requires that Y be a 4 x 1 vector.");
        }
        /* 为返回变量分配内存空间 */
        YP_OUT = mxCreateDoubleMatrix(m, n, mxREAL);
        /* 将输入输出参数指定指针变量 */
        yp = mxGetPr(YP_OUT);
        t = mxGetPr(T_IN);
        y = mxGetPr(Y_IN);
        /* 对数据进行计算 */
        yprime(yp,t,y);
        return;
      }
```

11.3.4 在 C 语言中使用由 MatLab 编写的函数和自定义函数

在 C 语言中可以使用 MatLab 的函数，如 M 函数、DLL 函数、MEX 函数以及内置函数，这些函数是通过 mexCallMatLab 函数来实现的。为了便于理解此过程，可仔细阅读下面的例子。此例中，函数调用 MatLab 的 eig 函数对矩阵进行特征值分解，并将分解所得结果存入 MAT 文件中。

```
#include "mex.h"
```

```
#include <math.h>
#include <mat.h>
void mexFunction(int nlhs,mxArray *plhs[],int nrhs, const mxArray
*prhs[])
{
MATFILE *fp;
Double ar[]={2,3,5,7};
Double ai[]={2,7.5,6.3,2.8};
MxArray *ap,*oap[2],*iap[2];
Int no,ni;
Ap=mxCreateDoubleMatrix(2,2,mxCOMPLEX);
Memcpy(mxGetPr(ap),pr,4*sizeof(double));
Memcpy(mxGetPi(ap),pi,4*sizeof(double));
/*用两个输出参数执行 eig 命令*/
No=2;ni=1;
Iap[0]=ap;
MexCallMatLab(no,cap,ni,iap,"eig");
Fp=matOpen("data.mat","w");
MxSetName(oap[0],"E");
MxSetName(oap[1],"V");
MatPutArray(fp,oap[0]);
/*用一个输出参数执行 eig 命令*/
No=1;ni=1;
MexCallMatLab(no,oap,ni,iap,"eig");
MxSetName(oap[0],"e");
MatPutArray(fp,oap[0]);
MatClose(fp);
MxDestroyArray(ap);
Exit(0);
}
```

11.4 习　　题

试分别用 C 和 FORTRAN 语言实现下面各题。

(1) 将 MAT 文件 A.mat 中的矩阵变量 A 读出，将其各元素的值乘以 2，并以名 B 存入新的 MAT 文件 B.mat 中，同时将变量 A 删除。

(2) 在 C 或 FORTRAN 程序中调用 MatLab 命令生成下面的矩阵，并将其网格图画出，停留 10 秒，然后自动关闭。

```
A =
     1     1     1     1     1
     2     4     8    16    32
     3     9    27    81   284
     4    16    64   256  1024
     5    25   125   625  3125
```

(3) 编写一个 MEX 函数，实现矩阵的高斯消元。

(4) 编写一函数，调用 MatLab 的 inv()函数来实现对矩阵求逆。

(5) 重新编写第 8 章中给出的求数值积分和数值微商的 MEX 函数，分别执行两种函数，了解它们的执行速度的差别。

附录 A　常用命令与函数

表 A.1　一般命令和演示函数

命令分类	函数指令	含　义
管理命令和函数目录	HELP	提供在线帮助目录
	WHAT	显示M，MAT和MEX文件
	TYPE	显示M文件的内容
	LOOKFOR	通过HELP入口的关键词检索
	WHICH	定位函数和文件的位置
	DEMO	运行演示程序的目录与实例
	PATH	控制MatLab的搜索路径
管理变量和工作空间	WHO	显示工作空间中的当前变量
	WHOS	以长形式显示当前工作空间中的变量
	LOAD	从磁盘上装载文件
	SAVE	将当前工作空间中的指定变量存入MAT文件中
	CLEAR	从内存中清除变量和函数
	SIZE	获得矩阵的大小
	LENGTH	得到向量的长度
	DISP	显示矩阵或文件
文件和系统管理函数	CD	改变当前工作目录
	DIR	显示当前工作目录下的目录和文件
	DELETE	删除文件
	!	执行操作系统的命令
	UNIX	执行操作系统的命令并返回结果
	DIARY	保存MatLab的交互文本
控制命令窗口操作	CEDIT	设置线编辑命令，调用设备管理参数
	CLC	清除命令窗口中的字符
	HOME	将光标返回初始位置
	FORMAT	设置输出格式
	ECHO	文本文件返回命令
	MORE	控制命令窗分页输出
	QUIT	推出MatLab工作空间
	STARTUP	当MatLab被调用时执行M文件
	EDIT	调用MatLab文本编辑器

续表

命令分类	函数指令	含 义
一般信息	INFO	关于MatLab的信息和MATHWORKS公司的信息
	SUBSCRIBE	加入MatLab的预约用户
	HOSTID	MatLab主服务程序的识别号
	WHATSNEW	说明书中没有包含的新信息
	VER	MatLab，Simulink，Toolbox的版本信息

表 A.2 MatLab 的低级文件操作的函数

命令分类	函数指令	含 义
文件基本操作	FOPEN	打开文件
	FCLOSE	关闭文件
	FREAD	从文件中读取二进制数据
	FWRITE	向一个文件写入二进制数据
	FSCANF	从文件读取格式化数据
	FRPINTF	将格式化数据写入文件
	FGETL	从文件中读取一行数据，并放弃换行符
	FGETS	从文件中读取一行数据，并保持换行符
	FERROR	查询文件输入输出的错误信息
	FEOF	检查文件结束标志
	FSEEK	设置文件的位置指针
	FREWIND	将一个打开文件的指针反绕
	FTELL	得到文件指针的位置
	SPRINTF	将格式化数据写入字符串中
	SSCANF	从格式化的字符串中读入数据
	TEMPNAME	建立临时文件名
	TEMPDIR	返回已经存在的临时目录名
文件的I/O	WK1CONST	WK1记录定义
	WK1READ	读WK1文件
	WK1WRITE	在WK1格式化文件中写矩阵
	WK1WREC	写WK1记录头
	CSVREAD	从逗号间隔的格式化文件中写矩阵
	DIMREAD	从以ASCII码限界的文件中读一矩阵
	DIMWRITE	按ASCII码限界的文件格式写一矩阵

表A.3 运算符与操作符

命令分类	运 算 符	含 义
运算符与特殊函数	+	加法
	−	减法
	*	矩阵乘法
	.*	数组乘法
	^	矩阵乘方
	.^	数组乘方
	\	左除
	/	右除
	./	数组除法
	KRON	KRONECKER张量积
	:	冒号操作符
	(	括号
	[	括号
	.	小数点
	..	原始目录
	...	连续号
	,	逗号
	;	分号
	%	注释号
	'	转置符或引号
	=	赋值符号
	==	等于号
	>	大于号
	<	小于号
关系运算符	&	逻辑与
	|	逻辑或
	~	逻辑非
	XOR	逻辑异或
逻辑符号	EXIST	检查变量或函数释放存在
	ANY	确认是否任何一个向量元素为非零
	ALL	确认是否所有向量或矩阵的元素是否为非零
	FIND	寻找非零元素的下标
	ISNAN	确认是否为不定值
	ISINF	确认是否为无穷大元素
	ISEMPTY	确认是否为空矩阵

续表

命令分类	运 算 符	含 义
逻辑符号	ISSTR	确认是否为字符串
	ISGLOBAL	确认变量是否为全局变量
	FINITE	当含有有限元时，其值为真
	ISINF	当含有无限大的元时，其值为真
	ISREAL	当矩阵为实矩阵时，其值为真
	ISSPARE	当矩阵为稀疏矩阵时，其值为真

表 A.4 字符串函数

命令分类	函数指令	含 义
一般字符串函数	ABS	字符串的ASCII码
	SETSTR	将整数值转换为字符串
	ISSTR	确认变量是否为文本字符串
	BLANKS	空格符
	DEBLANK	删除字符串后面的空格
	STR2MAT	由独立的字符串形成文本矩阵
	EVAL	执行包含MatLab表达式的字符串
字符串比较	STRCMP	比较两个字符串
	FINDSTR	在其他字符串中寻找一个字符串
	UPPER	将字符串中的小写字符转换为大写
	LOWER	将字符串中的大写字符转换为小写
	ISLETTER	确认字符串中的字符是否为字母
字符与数字转换	NUM2STR	将数字转换为字符串
	INT2STR	将整数转换为字符串
	STR2NUM	将字符串转换为数字
	SPRINTF	在给定格式下，将数字转换为字符串
	SSCANF	在给定格式下，将字符串转换为数字
	HEX2NUM	将十六进制字符串转换为浮点数
	HEX2DEC	将十六进制字符串转换为十进制整数
	DEC2HEX	将十进制整数转换为十六进制字符串

表 A.5 基本的数学函数

命令分类	函数指令	含 义
三角函数	SIN	正弦函数
	SINH	双曲函数

续表

命令分类	函数指令	含 义
三角函数	ASIN	反正弦函数
	ASINH	反双曲函数
	COS	余弦函数
	COSH	双曲余弦函数
	ACOS	反余弦函数
	ACOSH	反双余弦函数
	TAN	正切函数
	TANH	双曲正切函数
	ATAN	反正切函数
	ATANH	反双曲正切函数
	SEC	正割函数
	SECH	双曲正割函数
	ASEC	反正割函数
	ASECH	反双曲正割函数
	COT	余切函数
	COSTH	双曲余切函数
	ACOT	反余切函数
	ACOTH	反双曲余切函数
其他常用计算函数	FIX	超零方向取整
	ROUD	四舍五入到最近的整数
	FLOOR	超无穷方向取整
	REM	求两整数相除的余数
	CEIL	超正无穷方向取整
	EXP	求指数函数
	LOG	自然对数
	LOG10	求以10为底的函数
	SORT	求数值的平方根
	ABS	求绝对值
	CONJ	求复数的共轭
	IMAG	求复数的虚部
	REAL	求复数的实部

表 A.6 初等矩阵和矩阵变换

命令分类	函数指令	含 义
初等矩阵	ZEROS	零矩阵
	ONES	全一矩阵
	EYE	单位矩阵

续表

命令分类	函数指令	含　义
初等矩阵	RAND	均匀分布的随机矩阵
	RANDN	正态分布的随机矩阵
	LINSPACE	形成等间距的向量
	LOGSPACE	形成对数为等间距的向量
	MESHGRID	形成三维图形的X和Y数组
	:	形成正则空间矢量
矩阵变换	DIAG	产生或提取对角阵
	FLIPLR	将矩阵左右翻转
	FLIPUD	将矩阵上下翻转
	RESHAPE	改变矩阵的维数
	ROT90	将矩阵旋转N个90度
	TRIL	产生或提取下三角阵
	TRIU	产生或提取上三角阵
	:	提取部分矩阵或元素

表A.7　特殊变量和函数

函数指令	含　义
ANS	默认返回变量
EPS	默认的相对浮点精度
REALMAX	最大正浮点数，其值为1.7977E+308
REALMIN	最小正符点数，其值为2.2251E-308
PI	常量pi
I,J	虚数符号
INF	无穷值
NAN	不定值
FLOPS	浮点运算次数
NARGIN	函数的输入变量个数
NARGOUT	函数的输出变量个数
COMPUTER	本地计算机的类型
ISIEEE	判断计算机是否带有IEEE算法
VERSION	MatLab的版本信息
CLOCK	时钟
CPUTIME	CPU的实间(秒)
DATE	得到当天的日期信息
ETEME	得到计算机的运行时间
TIC	开始秒表的运行
TOC	读取秒表的运行时间

表 A.8 多项式函数

函数指令	含 义
ROOTS	多项式求根
POLY	求特征多项式
POLYVAL	求多项式的值
POLYVALM	求以矩阵为变量的多项式的值
RESIDUE	部分分式展开或留数计算
POLYFIT	多项式的曲线拟合
CONV	多项式的乘法
DECONV	多项式的除法
MKPP	建立分段多项式
PPVAL	计算分段多项式的值
RESI2	计算重极点的留数
UNMKPP	取消分段多项式
POLYDER	微分多项式

表 A.9 插值与查表

函数指令	含 义
INTERP1	一维数值插值
INTERP2	二维数值插值
INTERP3	三维数值插值
INTERFT	用快速FOURIER变换得到一维插值
GRINDDATA	数据网格
ICUBIC	一维数值函数的三此插值
SPLINE	三次样条数据插值
INTERPN	多维数值插值
MESHGRID	生成三维图形的X和Y矩阵
NDGRID	多维函数的坐标矩阵
TABLE1	一维表的查找
TABLE2	二维表的查找
XYZCHK	查找四维数据输入的变量

表 A.10 数据分析

函数指令	含 义
MAX	最大分量
MIN	最小分量
MEAM	平均或平均值
MEDIAN	中值
STD	标准偏差
SORT	将向量按升序排序
SUM	求元素的和
PROD	求元素的积
CUMSUM	求元素的累积和
CUMPROD	求元素的累积积
TRAPZ	利用梯形法求数值积分
DIFF	计算差分和近似微分
GRADIENT	计算近似梯度
DEL2	五点离散拉普拉斯变换
CROSS	向量的矢量积
DOT	向量的点积
SUBSPACE	求子空间之间的夹角

表 A.11 傅里叶变换

函数指令	含 义
CORRCOEF	相关系数
COV	协方差矩阵
SUBSPACE	子空间之间的夹角
FILTER	一维数字虑波
FILTER2	二维数字滤波
CONV2	二维卷积
FFT	离散傅立叶变换
FFT2	二维离散傅立叶变换
IFFT	离散逆傅立叶变换
IFFT2	二维离散逆傅立叶变换
ABS	求模
ANGLE	相角
UNWRAP	在弧度上展开相角
FFTSHIFT	平移零点到频线中心
CPLXPAIR	把数字分类为复共轭对
NEXTPOW2	最靠近2的幂次

表 A. 12　非线性数值方法

函数指令	含　义
ODE23	用低阶法解常微分方程和方程组
ODE23P	用低阶法解常微分方程并给出结果的图形
ODE45	用高阶法解常微分方程
QUAD	用低阶法求数值积分
QUAD8	用高阶法求数值积分
FMIN	求单变量函数的极小值
FMINS	求多变量函数的极小值
FZERO	求单变量函数的零点
FOPTIONS	定义优化过程中所使用的参数
QUAD8STP	QUAD8中所使用的递归函数
QUADSTP	QUAD函数中所使用的递归函数

表 A. 13　图形函数

命令分类	函数指令	含　义
基本的图形函数	PLOT	绘制向量或矩阵的图形
	PLOT3	三维空间内画点和直线的图
	FILL3	在三维空间内填画多边形
	LOGLOG	用全对数坐标绘制图形
	SEMILOGX	用X轴为对数的半对数坐标绘图
	SEMILOGY	用Y轴为对数的半对数坐标绘图
	FILL	绘制二维多边形填充图
特殊的图形函数	POLAR	绘制极坐标图
	BAR	画条形图
	STEM	画离散序列数据图
	STAIRS	画阶梯图
	ERRORBAR	画误差条形图
	HIST	画直方图
	ROSE	画玫瑰线或角度直方图
	COMPASS	画区域图
	FEATHER	画箭头图
	FPLOT	画任意函数的图形
	CONTOUR	画等高线图
	CONTOUR3	画三维等高线图
	CLABEL	在等高线图上增加高度标记

续表

命令分类	函数指令	含 义
特殊的图形函数	CONTOURC	等高线图计算
	PCOLOR	画伪色彩图形
	QUIVER	画箭头图
	MESH	画三维网状表面图
	MESHC	网络与等高图混合图形
	MESHZ	带参考平面的三维网格图
	SURF	三维表面阴影图
	SURFC	画表面阴影图与等高线图的混合图形
	SURFL	具有高度的三维表面阴影图
	WATERFALL	画落差图
	SLICE	画体积剖分图
	COMET	画彗星图
	COMET3	三维彗星图
三维图形的形式	VIEW	给三维图形指定观察点
	VIEWMTX	产生视点变换矩阵
	HIDDEN	网格消隐线移动方式
	SHADING	彩色阴影方式
	AXIS	控制坐标刻度和坐标的形式
	CAXIS	控制伪色彩坐标刻度
	COLORMAP	设置颜色查询表
	SLICE	立体可视图
	CYLINDER	产生圆柱体
	SPHERE	产生球体
图形的标注与注释	TITLE	为图形添加标题
	XLABEL	为X轴做文本标记
	YLABEL	为Y轴做文本标记
	ZLABEL	为Z轴做文本标记
	TEXT	在指定位置做文本注释
	GTEXT	用鼠标放置文本
	GRID	对二维和司维图形加栅格

表 A.14 常用图形控制函数

命令分类	函数指令	含 义
图形窗口和目标的建立与控制	FIGURE	建立新的图形窗口
	GCF	取消对当前图形的控制柄
	CLF	删除当前的图形对象

续表

命令分类	函数指令	含 义
图形窗口和目标的建立与控制	CLOSE	关闭指定的图形窗口
	AXES	建立坐标系
	LINE	建立直线对象
	TEXT	向当前图形写入文本
	PATCH	补充图形
	SURFACE	建立表面图形对象
	IMAGE	建立映象
	UICONTROL	建立用户界面控制
	UIMENU	建立用户界面菜单
	SET	设定图形目标的性质
	GET	取得图形目标的性质
	RESET	重新设置坐标系或图形
	DELETE	删除一个文件或图形目标
	DRAWNOW	填充未完成的图形事件
	NEWPLOT	为下一个图形性质操作而定义的M文件段落标记
坐标轴的控制	SUBPLOT	在指定的位置建立坐标系
	AXES	在任意的位置建立坐标系
	GCA	取得对当前坐标系的控制
	CLA	取消对当前坐标系的控制
	AXIS	控制坐标系的刻度和形式
	CAXIS	控制伪彩色坐标刻度
	HOLD	保持当前的图形
用户界面和对话框	DIALOG	用主对话框建立M文件
	FIGFLAG	当图形为当前显示时其值为真
	LAYOUT	定义对话框布局参数
	UIGUIDE	关于用户界面的约定、标准、建议的说明
	ERRORDLG	建立出错对话框
	HELPDLG	建立帮助对话框
	QUESTDLG	建立提问对话框
	WARNDLG	建立警告对话框
	UIPUTFILE	为保存文件提供对话框
	UIGETFILE	提供询问文件名的对话框
	UISETCOLOR	提供返回RGB颜色值的对话框
	UISETFONT	提供询问字体的对话框
打印控制	PRINT	打印或保存图形对象
	PRINTOPT	设定打印机的默认值
	ORIENT	设定当前图形在纸中的位置和取向

续表

命令分类	函数指令	含　义
打印控制	PRTPS	PostScript打印机驱动程序
	PRTWIN	MS Windows驱动程序
	CAPTURE	屏幕抓取当前的图形
其他图形操作	MOVIEIN	为存储动画画面而初始化内存
	GETFRAM	取得动画画面
	MOVIE	运行已经记录的动画画面
	GINPUT	由鼠标或游标输入图形
	RBBOX	涂抹块
	ISHOLD	返回HOLD状态
	WHITEBG	为白色背景设置图形默认值
	GRAYMON	为灰度显示器设置图形默认值
	TERMINAL	设置图形的终端类型
	ZOOM	二维图形的放大和缩小
	WAITFORBUTTONPRESS	在图形中等待按建

表 A.15　色彩和亮度控制函数

命令分类	函数指令	含　义
色彩控制及映象表操作	COLORMAP	设置颜色查找表
	CAXIS	伪色彩坐标刻度
	SHADING	彩色阴影方式
	HSV	色彩一饱和度映像表
	GRAY	线性灰度变换表
	HOT	黑一红一黄一白4色映像表
	COOL	深蓝一品红2色浓淡映像表
	BONE	以蓝色为基调的浓淡映像表
	COPPER	线性青铜色调映像表
	PINK	淡粉红色浓淡映像表
	JET	一种HSV的变体
	FLAG	红一白一蓝一黑交替变换的颜色映像表
	HSV2RGB	色度图向红、绿、蓝映像表的转换
	RGB2HSV	红、绿、蓝映像表向色度图转换
	CONTRAST	灰度图像的对比度增强
	BRIGHTEN	增亮和变暗当前色彩的映像表
	RGBPLOT	绘制彩色图形
	SPINMAP	彩色映像表的旋转
	PRISM	光谱颜色板
	COLORBAR	显示颜色条

续表

命令分类	函数指令	含 义
亮度控制	SURFL	具有亮度的三维表面阴影图
	SPECULAR	镜面反射率
	DIFFUSF	漫反射率
	SURFNORM	三维表面的法线

表 A.16 信号处理功能

函数指令	含 义
SOUND	将向量转化为声信号
SAXIS	将声轴比例化
AUWRITE	写已按MU-LAW编码的声音文件
AUREAD	读已按MU-LAW编码的声音文件
MU2LIN	将MU-LAW编码的声音信号变成线性信号
LIN2MU	将线性信号变成MU-LAW编码的声音文件

表 A.17 特殊矩阵函数

函数指令	含 义
COMPAN	友矩阵
GALLERY	几个小的实验矩阵
HADAMARD	HADAMARD矩阵
HANKEL	HANKEL矩阵
HILB	HILBERT矩阵
INVHILB	逆HILBERT矩阵
MAGIC	魔方阵
PASCAL	PASCAL矩阵
ROSSER	经典对称特征值实验问题
TOEPLITZ	TOEPLITZ矩阵
VANDER	VANDERMONDE矩阵
WILKISON	WILKINSON特征值测试矩阵

表 A.18 稀疏矩阵函数

函数指令	含 义
SPEYE	稀疏单位阵
SPRANDN	随机稀疏矩阵
SPRANDSYM	随机稀疏对称阵

续表

函数指令	含 义
SPDIAGS	取出并生成稀疏带和对角阵
SPARSE	由非零元素和下标建立稀疏矩阵
FULL	将稀疏矩阵变成满阵
FIND	寻找非零元素的下标
SPCONVERT	从稀疏矩阵外结构的变换
NNZ	非零元素的数目
NONZEROS	寻找非零元素
NZMAX	为非零元素分配的内存数
SPONES	用1代替非零元素
SPALLOC	为非零元素分配内存空间
ISSPARSE	判断矩阵是否为稀疏矩阵
SPFUN	只对非零元素进行函数运算
SPY	显示稀疏结构
GPLOT	按图论画图
COLMMD	列最小度
SYMMMD	最小对称度
SYMRCM	逆CUTHILL-MCKEE序
RANDPERM	随机排列向量
DMPERM	Dulmage-Mendelsohn分解
NORMEST	计算二阶范数
CONDEST	求一阶范数条件数
SPRANK	稀疏矩阵的秩
TREELAYOUT	显示一个或多个结构树
TREEPLOT	画结构树的图
ETREE	给出矩阵的消元树
ETREEPLOT	画消元树的图
SYMBFACT	符号分解分析
SPPPARMS	为稀疏矩阵处理过程设置参数
SPAUGMENT	形成最小二乘增广系统
RJR	随机JACOBI旋转变换
UHMESH	将一组网络边界变换成图形和矩阵

表 A.19 特殊矩阵函数

函数指令	含 义
GAMMA	伽玛函数
GAMMAINC	非完全伽玛函数

续表

函数指令	含 义
GAMAALN	对数伽玛函数
BESSELJ	第一类贝塞尔函数
BESSELY	第二类贝塞尔函数
BESSELI	改良型第一类贝塞尔函数
BESSELK	改良型第二类贝塞尔函数
BESSELA	分数阶贝塞尔函数和精度估计
BETA	贝塔函数
BETAINC	非完全贝塔函数
BETALN	对数贝塔函数
BETACORE	用于非完全贝塔函数的核心算法
ELLIPJ	雅可比椭圆函数
ELLIPKE	完全椭圆积分
ERF	误差函数
ERFC	互补误差函数
ERFCX	比例互补误差函数
ERFINV	逆误差函数
GCD	最大公约数
LCM	最小公倍数
LOG2	分割浮点数
POW2	比例浮点数
RAT	有理逼近
RATS	有理输出
CART2POL	改变笛卡尔坐标系为极坐标系
POL2CART	改变极坐标系为笛卡尔坐标系
CART2SPH	改变笛卡尔坐标系为球坐标系
SPH2CART	改变球坐标系为笛卡尔坐标系

表 A.20 线性代数特殊矩阵函数

命令分类	函数指令	含 义
线性代数计算	/、\	求线性方程组的解
	CHOL	Cholesky分解
	INV	求矩阵的逆
	LU	lu分解，即高斯消元法求系数矩阵
	QR	QR分解，即正交三角分解
	NNLS	非负最小二乘解
	PINV	矩阵的伪逆
	QRDELETE	从QR分解中去掉一列

续表

命令分类	函数指令	含　义
线性代数计算	QRINSERT	在QR分解中插入一列
	LSCOV	在协方差已知的情况下求最小二乘解
矩阵分析函数	NORM	计算矩阵和向量的范数
	RCOND	Linpack逆条件值估计
	COND	计算矩阵的条件数
	RANK	计算矩阵的秩
	DET	计算矩阵的行列式的值
	TRACE	计算矩阵的迹
	ORTH	正交化
	RREF	减缩行列式矩阵
特征值和奇异值函数	EIG	求矩阵的特征值和特征向量
	POLY	求矩阵的特征多项式
	POLYEIG	求多项式的特征值
	HESS	Hessberg形式
	QZ	求矩阵的广义特征值
	RSF2CSF	变实分块对角阵为复对角阵
	CDF2RDF	变复对角阵为实分块对角阵
	SCHUR	Schur分解
	BALANCE	对矩阵进行均衡处理以提高求特征值的精度
	SVD	对矩阵进行奇异值分解
矩阵函数	EXPM	矩阵指数
	EXPM1	实现EXPM的M文件
	EXPM2	通过泰勒级数求矩阵的指数
	EXPM3	通过特征值和特征向量来求矩阵的指数
	LOGM	矩阵的对数
	SQRTM	对矩阵进行开平方
	FUNM	一般矩阵的计算

表 A. 21　语言结构与程序调试函数

命令分类	函数指令	含　义
MatLab的编程语言	SCRIPT	MatLab的原始文件和M文件
	FUNCTION	定义MatLab的M函数
	EVAL	执行带有MatLab表达式的字符串
	FEVAL	执行由字符串定义的函数
	GLOBAL	定义全局变量
	NARGCHK	检查输入变量的个数
	LASTERR	最后一个错误的错误信息

续表

命令分类	函数指令	含 义
程序控制流	IF	条件执行语句
	ELSE	与IF一起使用
	ELSEIF	与IF一起使用
	END	FOR、WHILE、IF语句的结束标志语句
	FOR	有一定循环次数的循环语句
	WHILE	无规定循环次数的循环语句
	BREAK	中断内层的循环
	RETURN	返回调用函数
	ERROR	显示错误信息和故障信息
	WARNING	显示警告信息
交互式输入	INPUT	提示用户输入
	KEYBOARD	请求键盘作为一个原始文件的输入方式
	MENU	产生一个供用户选择的菜单
	PAUSE	等候用户响应
	UIMENU	建立一个用户界面菜单
	UICONTROL	建立一个用户界面控制
调试命令	DBSTOP	设置中断点
	DBCLEAR	删除中断点
	DBCONT	重新开始运行
	DBDOWN	改变当前的工作空间
	DBSTACK	列表显示堆栈调用
	DBSTATUS	列表显示所有中断点
	DBSTEP	执行一行或多行
	DBTYPE	列表表示带有行号的M文件
	DBUP	改变当前的工作空间
	DBQUIT	推出DEBUG模式
	NEXDEBUG	调试MEX文件

表 A. 22 音频处理函数

函数指令	含 义
SOUND	变向量为音频信号
SAXIS	音频轴刻度
AUWRITE	按WU-LAW写编码的音频文件
AUREAD	按WU-LAW读编码的音频文件
WAVWRITE	写MS
WAVREAD	读MS
MU2LIN	变WU-LAW编码的音频文件外2线性音频信号
LIN2MU	变线性音频信号为WU-LAW编码的音频信号

附录B TOOLBOX 函数

MatLab 系统提供了许多 TOOLBOX(工具箱)，而且由于 MatLab 的可扩充性，TOOL-BOX 的数目与日俱增。

表 B.1 局部函库

函数指令	含 义
MatLabRC	MatLab的主启动M文件
PRINTOPT	设置打印选项

表 B.2 信号处理工具箱

命令分类	函数指令	含 义
波形产生	DIRIC	产生DIRICHLET或周期SINC函数
	SAWTOOTH	产生锯齿波或三角波
	SINC	产生SINC或SIN(ΠT)ΠT函数
	SQUARE	产生方波
滤波器分析和实现	ABS	取绝对值(幅值)
	ANGLE	取相角
	CONV	求卷积
	FILTER	直接滤波器实现
	FILTFILT	零相位数字滤波
	FILTIC	FILTER函数初始条件选择
	FLTFILT	重叠相加法FFT滤波器实现
	FREQS	模拟滤波器频率响应
	FREQSPACE	频率响应中的频率间隔
	FREQZ	数字滤波器频率响应
	GRPDELAY	平均滤波延迟(群延迟)
	IMPZ	数字滤波器的冲激响应
	ZPLANE	离散系统零极点图
线性系统变换	CONVMTX	卷积矩阵
	POLY2RC	从多项式系统中计算反射系数
	RC2POLY	从反射系数中计算多项式系数
	RESIDUEZ	Z变换部分分式展开或留数计算

续表

命令分类	函数指令	含　义
线性系统变换	SOS2SS	变系统二阶分割形式为状态空间形式
	SOS2TF	变系统二阶分割形式为传递函数形式
	SOS2ZP	变系统二阶分割形式为零极点增溢形式
	SS2SOS	变系统状态空间形式为二阶分割形式
	SS2TF	变系统状态空间形式为传递函数形式
	SS2ZP	变系统状态空间形式为零极点增溢形式
	TF2SS	变系统传递函数形式为状态空间形式
	TF2ZP	变系统传递函数形式为零极点增溢形式
	ZP2SOS	变系统零极点增溢形式为二阶分割形式
	ZP2SS	变系统零极点增溢形式为状态空间形式
	ZP2TF	变系统零极点增溢形式为传递函数形式
IIR滤波器设计	BESSELF	BESSEL(贝塞尔)模拟滤波器设计
	BUTTER	BUTTERWORTH(比特沃思)滤波器设计
	CHEBY1	CHEBYSHEV(切比雪夫) I 型滤波器设计
	CHEBY2	CHEBYSHEV(切比雪夫) II 型滤波器设计
	ELLIP	椭圆滤波器设计
	YULEWALK	递归数字滤波器设计
IIR滤波器阶的选择	BUTTORD	BUTTERWORTH滤波器阶的选择
	CHEBLORD	CHEBYSHEV I 型滤波器阶的选择
	CHEB2ORD	CHEBYSHEV II 型滤波器阶的选择
	CLLIPORD	椭圆滤波器阶的选择
FIR滤波器设计	FIRL	基于窗函数的FIR滤波器设计——标准响应
	FIR2	基于窗函数的FIR滤波器设计——任意响应
	FIRLS	最小二乘FIR滤波器设计
	INTFILT	内插FIR滤波器设计
	REMEZ	PARKS-MCCELLAN最优FIR滤波器设计
	REMEZORD	PARKS-MCCELLAN最优FIR滤波器阶估计
变换	CZT	线性调频Z变换
	DCT	离散余弦变换(DCT)
	DFTMTX	离散傅里叶变换矩阵
	FFT	一维快速傅里叶变换
	FFTSHIFT	重新排列FFT的输出
	HILBERT	HILBERT(希尔伯特)变换
	IDCT	逆离散余弦变换
	IFFT	一维逆快速傅里叶变换
统计信号处理	CORRCOEF	相关系数矩阵
	COV	协方差矩阵
	XCOV	互协方差函数估计

续表

命令分类	函数指令	含 义
统计信号处理	COHERE	相关函数平方幅值估计
	CSD	互谱密度(CSD)估计
	PSD	信号功率谱密度(PSD)估计
	TFE	从输入输出中估计传递函数
	XCORR	互相关函数估计
窗函数	BARTLETT	BARTLETT(巴特利特)窗
	BLACKMAN	BLACKMAN(布莱克曼)窗
	BOXCAR	矩形窗
	CHEBWIN	CHEBYSHEV窗
	HAMMING	HAMMING(哈明)窗
	HANNING	HANNING(汉宁)窗
	KAISER	KAISER窗
	TRIANG	三角窗
参数化建模	INVFREQS	模拟滤波器拟合频率响应
	INVFREQZ	离散滤波器拟合频率响应
	LEVINSON	LEVINSON-DURBIN递归算法
	LPC	维性预测系数
	PRONY	利用PRONY方法的离散滤波器拟合时间响应
	STMCB	利用STEIGLITZ-MCBRIDE迭代方法求线性模型
特殊操作	CCEPS	例谱分析和最小相位重构
	DECIMATE	降低序列的取样速率
	DECONV	反卷积和多项式除法
	DEMOD	通讯仿真中的解调
	INTERP	提高取样速率(内插)
	MEDFILT1	一维中值滤波
	MODULATE	通讯仿真中的调制
	RCEPS	实例谱和最小相位重构
	RESAMPLE	改变取样速率
特殊操作	SPECGRAM	频谱分析
	VCO	电压控制振荡器
模拟原型滤波器设计	BESSELAP	BESSEL模拟低通滤波器原型
	BUTTAP	BUTTERWORTH模拟低通滤波器原型
	CHEB1AP	CHEBYSHEV Ⅰ 型模拟低通滤波器原型
	CHEB2AP	CHEBYSHEV Ⅱ 型模拟低通滤波器原型
	ELLIPAP	椭圆模拟低通滤波器原型
频率变换	LP2BP	低通到带通模拟滤波器变换
	LP2HP	低通到高通模拟滤波器变换

续表

命令分类	函数指令	含 义
频率变换	LP2BS	低通到带阻模拟滤波器变换
	LP2LP	低通到低通模拟滤波器变换
滤波器离散化	BILINEAR	双线性变换
	IMPINVAR	冲激响应不变法实现模拟到数字的滤波器变换
其他	CONV2	二维卷积
	CPLXPAIR	将复数归成复共轭对
	DETREND	删除线性趋势
	FFT2	二维快速傅里叶变换
	FILTER2	二维数字滤波器
	IFFT2	二维逆快速傅里叶变换
	POLYSTAB	稳定多项式
	XCORR2	二维互相关

表B.3 图像处理工具箱

命令分类	函数指令	含 义
图像输入/输出	BMPREAD	从磁盘中读BMP(MICROSOFT WINDOWS下的位图)文件
	BMPWRITE	将一BMP文件写入到磁盘
	GIFREAD	从磁盘中读GIF文件
	GIFWRITE	将GIF文件写入磁盘
	HDFPEEK	在HDF文件中列出目标标记 / 参考对
	HDFREAD	从HDF文件中读取数据
	HDWRITE	写数据到HDF文件中
	PCXREAD	从磁盘中读PCX文件
	PCXWRITE	将PCX文件写入磁盘
	TIFFREAD	从磁盘中读TIFF文件
	TIFFWRITE	将TIFF文件写入磁盘
	XWDREAD	从磁盘中读XWD文件
	XWDWRITE	将XWD文件写入磁盘
实用程序	GETIMAGE	从坐标系中读取图像数据
	ISBW	当图像为黑白图像时，其值为真
	ISGRAY	当图像为灰度图像时，其值为真
	ISIND	当图像为加标图像时，其值为真
颜色操作	BRIGHTEN	加亮或增暗一颜色板
	CMUNIQUE	寻找唯一的颜色板及相应的图像置换颜色板位置

续表

命令分类	函数指令	含　义
颜色操作	CMPERMUTE	置换颜色板位置
	CMGAMMA	R校正颜色板
	CMGAMDEF	默认的R校正表
	DITHER	FLOYD-STEINBERG图像颤抖算法
	HSV2RGB	变HSV值为RGB颜色空间
	IMADJUST	调整并增强图像强度
	IMAPPROX	利用更少颜色的图像逼近加标图像
	NTSC2RGB	变NTSC值为RGB颜色空间
	RGB2GRAY	变RGB值为灰度值
	RGB2HSV	变RGB值为HSV颜色空间
	RGB2NTSC	变RGB值为NTSC颜色空间
	RGBPLOT	绘制RGB颜色板分量的图形
几何操作	IMCROP	修剪图像
	IMRESIZE	改变图像大小
	IMROTATE	旋转图像
	TRUESIZE	改变图像大小使之具有实际尺寸
	IMZOOM	放大或缩小图像和二维图形
图像增强/分析	BRIGHTEN	增强或削弱颜色板
	GRAYSLICE	密度(强度)限幅
	HISTEQ	直方图均衡化
	IMADJUST	调整和展宽图像强度
	IMAPPROX	利用较少颜色的图像逼近图像
	IMHIST	图像直方图
	IMPIXEL	一像素点的颜色
	IMPROFILE	轮廓强度
	INTERP2	二维数据内插
图像统计	CORR2	二维相关系数
	MEAN2	矩阵的均值
	STD2	二维标准差
形态操作	BWAREA	二进制图像中的目标区域
	BWEULER	欧拉数
	BWMORPH	形态算子
	BWPERIM	二进制图像中目标的周围
	DILATE	加浓二进制图像
	EDGE	边界提取
	ERODE	冲淡二进制图像

续表

命令分类	函数指令	含 义
FIR(有限冲激响应)滤波器设计	FSAMP2	通过频率取样的二维FIR滤波器设计
	FSPECIA1	特殊的二维滤波器
	FTRANS2	通过频率变换的二维FIR滤波器设计
	FWIND1	使用一维窗函数的FIR滤波器设计
	FWIND2	使用二维窗函数的FIR滤波器设计
	IMNOISE	图像噪声
频率响应	FREQSPACE	二维频率响应的频率空间
	FREQZ2	二维频率响应
滤波	COLFILT	局部非线性滤波
	CONV2	二维卷积
	FILTER2	二维滤波
	MEDFILT2	二维中值滤波
	MFILTER2	屏蔽滤波
	NLFILTER	局部非线性滤波
	WIENER2	自适应二维维纳滤波
分块处理	BESTBLD	分块处理的最佳块大小
	BLKPROC	按块处理一图像
	COL2IM	重新排列以形成图像
	COLFILT	局部非线性滤波
	IM2COL	重新排列成列
个别区域	MFILTER2	屏蔽滤波
	ROIPOLY	定义感兴趣的多边区域
	ROICOLOR	用颜色定义感兴趣的多边区域
变换	DCT2	二维离散余弦变换
	FFT2	二维快速傅里变换
	FFTSHIFT	零频移到频谱中心
	IDCT2	二维逆离散余弦变换
	IFFT2	二维逆快速傅里叶变换
	RADON	RADON变换
	DITHER	FLIYD-STEINBERG图像颤抖
	GRAY2IND	变灰度图像为附标图像
	HSV2RGB	变HSV值为RGB值
	IM2BW	变图像为黑白图形
	IMSLICE	在图像中获取／置入图像块
	IND2GRAY	变附标图像为灰度图像
	IND2RGB	变附标图像为RGB图像
	MAT2GRAY	变矩阵为(灰度)图像

续表

命令分类	函数指令	含　义
变换	NTSC2RGB	变NTSC值为RGB值
	RGB2GRAY	变RGB图像或值为灰度图像或值
	RGB2HSV	变RGB值为HSV值
	RGB2IND	变RGB图像为附标图像
	RGB2NTSC	变RGB值为NTSC值
图像显示	COLORBAR	显示颜色条
	COLORMAP	设置或获取颜色查找表
	GRAY	线性灰度颜色板
	HSV,HOT,JET	颜色板
	IMAGE	显示附标图像
	IMAGESC	数据定标并按图像显示
	IMCONTOUR	图像等高线
	IMMOVIE	制作图像动画
	IMSHOW	显示所有类型的图像数据
	MONTAGE	按矩形剪辑方式显示图像
	SUBIMAGE	显示多个图像
	WARP	将图像卷成曲面
演示	IMDEMO	一般图像处理演示
	DCTDEMO	二维离散余弦变换图像压缩演示
	FIRDEMO	二维FIR滤波器演示
	NIFDEMO	二维非线性滤波演示
专用函数	CUMSUM3D	三维矩阵封装成二维矩阵时的累积和
	DCT	一维离散余弦变换
	DETMTX2	一元二维离散余弦变换矩阵
	DITHERC	图像颤抖的MEX文件
	ELEM3D	三维矩阵封装成二维矩阵的元素位置
	GETLINE	利用橡皮线跟踪鼠标移动
	GETPTS	利用可视点跟踪鼠标移动
	GETRECT	利用橡皮矩形跟踪鼠标移动
	GIF	压缩GIF数据
	HDFPEEKC	搜索HDF文件的MEX文件
	HDFREADC	读HDF文件的MEX文件
	HDFWC	写HDF文件的MEX文件
	IDCT	一维逆离散余弦变换
	IM2GRAY	变图像为灰度
	IMHISTC	图像直方图计算的MEX文件
	NDX3D	三维矩阵封装成二维矩阵的索引

续表

命令分类	函数指令	含 义
专用函数	RGB2IM	变RGB图像为附标或强度图像
	RLE	压缩编码数据
	TIFF	压缩TIFF编码数据
	VMQUANT	与彩色量化MEX文件接口的M文件
	WAITBAR	显示等待条
MAT文件	BWMORPH.MAT	BWMORPH.M文件的查找表
	FOREST.MAT	CARMANAH OLD GROWTH FOREST的扫描相片
	MRI.MAT	人体心脑的磁性共振图像
	TREES.MAT	树的扫描图像

表 B.4 控制系统工具箱

命令分类	函数指令	含 义
建模	APPEND	追加系统动态特性
	ARD2	产生二阶系统的A，B，C，D
	AUGSTATE	变量状态作为输出
	BLKBUILD	从方框图中构造状态空间系统
	CLOOP	系统的闭环
	CONNECT	方框图建模
	CONV	两个多项式的卷积
	DESTIM	从增溢矩阵中形成离散状态估计器
	DREG	从增溢矩阵中形成离散控制器和估计器
	DRMODEL	产生随机离散模型
	ESTIM	从增溢矩阵中形成连续状态估计器
	FEEDBACK	反馈系统连接
	PADE	时延的PADE近似
	PARALLEL	并行系统连接
	REG	从增溢矩阵中形成连续控制器和估计器
	RMODEL	产生随机连续模型
	SERIES	串行系统连接
	SSDELETE	从模型中删除输入、输出或状态
	SSSELECT	从大系统中选择子系统
模型变换	C2D	变连续系统为离散系统
	C2DM	利用指定方法变连续为离散系统
	C2DT	带一延时变连续为离散系统
	D2C	变离散为连续系统
	D2CM	利用指定方法变离散为连系统

续表

命令分类	函数指令	含 义
模型变换	POLY	变根值表示为多项式表示
	RESIDUE	部分会式展
	SS2TF	变状态空间表示为传递函数表示
	SS2ZP	变状态空间表示为零极点表示
	TF2SS	变传递函数表示为状态空间表示
	TF2ZP	变传递函数表示为零极点表示
	ZP2TF	变零极点表示为传递函数表示
	ZP2SS	变零极点表示为状态空间表示
A362模型简化	BALREAL	平衡实现
	DBALREAL	离散平衡实现
	DMODRED	离散模型降阶
	MINREAL	最小实现和零极点对消
	MODRED	模型降阶
模型实现	CANON	正则形式
	CTRBF	可控阶梯形
	OBSVF	可观阶梯形
	SS2SS	采用相似变换
A373模型特性	COVAR	相对于白噪声的连续协方差响应
	CTRB	可控性矩阵
	DAMP	阻尼系统和固有频率
	DCGAIN	连续稳态(直流)增益
	DCOVAR	相对于白噪声的离散协方差响应
	DDAMP	离散阻尼系统和固有频率
	DDCGAIN	离散稳态(直流)增益
	DGRAM	离散可控性和可观性
	DSORT	按幅值排序离散特征值
	EIG	特征值和特征向量
	ESORT	按实部排序连续特征值
	GRAM	可控性和可观性
	OBSV	可观性矩阵
	PRINTSYS	按格式显示系统
	ROOTS	多项式之根
	TZERO	传递零点
	TZERO2	利用随机扰动法传递零点
时域响应	DIMPULSE	离散时间单位冲激响应
	DINITIAL	离散时间零输入响应
	DLSIM	任意输入下的离散时间仿真

续表

命令分类	函数指令	含　义
时域响应	DSTEP	离散时间阶跃响应
	FILTER	单输入单输出Z变换仿真
	IMPULSE	冲激响应
	INITIAL	连续时间零输入响应
	LSIM	任意输入下的连续时间仿真
	LTITR	低级时间响应函数
	STEP	阶跃响应
	STEPFUN	阶跃函数
A403频域响应	BODE	BODE(波特)图(频域响应)
	DBODE	离散BODE图
	DNICHOLS	离散NICHOLS图
	DNYQUIST	离散NYQUIST图
	DSIGMA	离散奇异值频域图
	FBODE	连续系统的快速BODE图
	FREQS	拉普拉斯变换频率响应
	FREQZ	Z变换频率响应
	LTIFR	低级频率响应函数
	MARGIN	增益和相位裕度
	NICHOLS	NICHOLS图
	NGRID	画NICHOLS图的栅格线
	NTQUIST	NYQUIST图
	SIGMA	奇异值频域图
根轨迹	PZMAP	零极点图
	RLOCFIND	交互式地确定根轨迹增益
	RLOCUS	画根轨迹
	SGRID	在ωN，Z网格上画连续根轨迹
	ZGRID	在ωN，Z网格上画离散根轨迹
增益选择	ACHKER	单输入单输出极点配置
	DLQE	离散线性二次估计器设计
	DLQEW	离散线性二次估计器设计
	DLQR	离散线性二次调节器设计
增益选择	DLQRY	输出加权的离散调节器设计
	LQE	线性二次估计器设计
	LQED	基于连续代价函数的离散估计器设计
	LQE2	利用SCHUR法设计线性二次估计器
	LQEW	一般线简二次估计器设计
	LQR	线性二次调节器设计

续表

命令分类	函数指令	含 义
增益选择	LQRD	基于连续代价是函数的离散调节器设计
	LQRY	输出加权的调节器设计
	LQR2	利用SCHUR法设计线性二次调节器
	PLACE	极点配置
方程求解	ARE	代数RICCATI方程求解
	DLYAP	离散LYAPUNOV方程求解
	LYAP	连续LYAPUNOV方程求解
	LYAP2	利用对角化求解LYAPUNOV方程
演示示例	BOILDEMO	锅炉系统的LQG设计
	CTRLDEMO	控制工具箱介绍
	DISKDEMO	硬盘控制器的数字控制
	JETDEMO	喷气式飞机偏航阻尼的典型设计
	KALMDEMO	KALMAN滤波器设计和仿真
实用工具	ABCDCHK	检测(A，B，C，D)组的一致性
	CHOP	取N个重要的位置
	DEXRESP	离散取样响应函数
	DFRQINT	离散BODE图的自动定范围的算法
	DFRQINT2	离散NYQUIST图的自动定范围的算法
	DMULRESP	离散多变量响应函数
	DISTS1	到直线间的距离
	DRIC	离散RICCATI方程留数计算
	DSIGMA2	DSIGMA实用工具函数
	DTIMVEC	离散时间响应的自动定范围算法
	EXRESP	取样响应函数
	FREQINT	BODE图的自动定范围算法
	FREQINT2	NYQUIST图的自动定范围算法
	FREQRESP	低级频率响应函数
	GIVENS	旋转
	HOUSH	构造HOUSEHOLDER变换
	IMARGIN	利用内插技术求增溢和相位裕度
	LAB2SER	变标号为字符串
	MULRESP	多变量响应函数
	NARGCHK	检测M文件的变量数
	PERPXY	寻找最近的正交点
	POLY2STR	变多项式为字符串
	PRINTMAT	带行列号打印矩阵
	RIC	RICCATI方程留数计算

续表

命令分类	函数指令	含 义
实用工具	SCHORD	有序SCHWR分解
	SINMA2	SIGMA实用工具函数
	TFCHK	检测传递函数的一致性
	TIMVEC	连续时间响应的自动定范围算法
	TZREDUCE	在计算过零点时简化系统
	VSORT	匹配两根轨迹的向量

表B.5 非线性控制设计工具箱

命令分类	函数指令	含 义
A483对话框管理	CONEDDLG	管理NCD工具箱固定编辑器的对话框
	PARAMDLG	管理NCD优化参数的对话框
	RANGEDLG	管理坐标系范围的对话框
	REFDLG	管理NCD参考信号的对话框
	STEPDLG	管理NCD阶跃响应的对话框
	UNCERDLG	管理NCD不确定变量的对话框
主要界面	CONTRNCD	建立NCD固定图形的用户界面控制
	MENUNCD	建立NCD固定图形的用户界面菜单
	NCDBLOCK	包含NCD框图的SIMULINK系统
	OPTBLOCK	打开一个NCD图形的底稿文件
	OPTFIG	建立一个NCD固定图形
主要优化	COSTFUN	NCD优化的代价函数
	NLINOPT	执行优化算法
演示示例	NCDDEMO	包含所有NCD演示示例的SLMULINK系统
	NCDDEMO1	PID控制器
	NCDDEMO2	带前馈控制器的LQR
	NCDDEMO3	多输入多输出的PI控制器
	NCDDEMO4	例摆演示
教程	NCDTUT1	控制设计示例
	NCDTUT2	系统辨识示例
用户界面工具	DIALOG	主对话框建立M文件
	ERRORDLG	建立出错对话框
	FIGFLAG	当图形为当前显示在屏幕上时，其值为真
	HELPDLG	显示一帮助对话框
	LAYOUT	定义对话框布局参数的底搞文件
	QUESTDLG	建立提问对话框
	UIGUIDE	有关用户界面约定／标准／建议的说明
	WARNDLG	建立警告对话框

续表

演示和教程实用工具	NCD1INIT	为NCDDEMO1的优化进行设置
	NCD2INIT	为NCDDEMO2的优化进行设置
	NCD3INIT	为NCDDEMO3的优化进行设置
	NCD4INIT	为NCDDEMO4的优化进行设置
	PENDDATA	为NCDTUT2(即倒摆)进行设置
界面实用工具	CUTOBJ	提供有关当前点的信息
	DIVIDECB	将固定界分为两部分
	DELLINE	从NCD图中删除所有的图
	DONEP	收回CLOSE按钮和菜单
	ERRORNCD	管理NCD产生的常见错误，它调用ERRORDLG(出错对话框)
	FILLAXES	建立约束边界并进行数据检测
	FORCEIT	在已存在的界限内插入一子集
	KEYNCD	NCD按键函数
	LOADNCD	装入并显示NCD数据
	MAKESURF	建立并限界曲面
	REFRESHO	使约束矩阵与图形一致
	SAVELOAD	当文件是从SELECTFILE中选择时，其值为真
	SNAPNCD	以22.5间隔排出约束条
	TEXTED	收回PORT可编辑的文本
	UNDONCD	放弃上次NCD图形用户界面的操作
	UPDATDLG	更新NCD对话框
最优化实用工具	CONVERTM	变约束矩阵为最优化格式
	MINIPARS	NCD最小化分析
	MONTEVAR	初始化MONTECARLO仿真
	NCDGLOB	定义NCD全局变量
	STR2MAT2	变一行字符串为多行字符串
帮助文本文件(以.HLP为扩展名)	HOTKEY	热键帮助
	MAINNCD	一般NCD帮助
	PARAMDLG	最优化参数对话框的帮助
	READNCD	与README.M文件内容相同
	STEPDLG	阶跃响应对话框的帮助
	UNCERDLG	不确定性变量对话框的帮助

表B.6 鲁棒控制工具箱

命令分类	函数指令	含 义
可选系统数据结构	BRANCH	从树中提取一分支
	GRAFT	在树中增加一分支
	ISSYSTEM	辨识一系统变量
	ISTREE	辨识一树型变量
	MKSYS	为系统建立树变量
	TREE	建立树变量
	VRSYS	返回标准系统变量名
建模	AUGSS	系统增广(状态空间模型)
	AUGTF	系统增广(传递函数模型)
	INTERR	一般多变量内连系统
模型转换	BILIN	多变量双线性变换
	DES2SS	利用奇异值分解变系统为状态空间系统
	LFTF	线性分式变换
	SECTF	扇形变换
	SLOWFAST	慢／快分解
	STABPROJ	稳定和逆稳定映射
	TFM2SS	变传递函数模型为状态空间模型
实用工具	ARESOLV	广义连续时间RICCATI方程求解
	BLKRSCH	通过CSCHUR得到块有序实SCHUR形式
	CSCHUR	通过复旋转得有序复SCHUR形式
	DARESOLV	广义离散时间RICCATI方程求解
	DRICCOND	离散时间RICCATI方程的条件数
	RICCOND	连续时间RICCATI方程的条件数
多变量BODE图	CGLOCI	连续特性增益轨迹
	DCGLOCI	离散特性增益轨迹
	DSIGMA	离散奇异值BODE图
	MUOPT	具有实／复数混合不确定性系统的SSV(结构化奇异值)上界
	OSBORNE	通过OSBORNE法求得的SSV上界
	PERRON	计算PERRON特征值
	PSV	PERRON特征结构的SSV
	SIGMA	连续奇异值BODE图
	SSV	结构化奇异值BODE图
因子分解技术	IOFC	内外因子分解(列类型)
	IOFR	内外因子分解(行类型)

续表

命令分类	函数指令	含　义
因子分解技术	SFL	左边频谱分解
	SFR	右边频谱分解
模型简化方法	BALMR	截断均衡模型简化
	BSTSCHML	相对误差SCHUR模型简化
	BSTSCHMR	相对误差SCHUR模型简化
	IMP2SS	从脉冲响应到状态空间实现
	OBALREAL	有序均衡实现
	OHKLMR	最优HANKEL极小化逼近
	RSCHUR	SCHUR模型简化
鲁棒控制综合方法	DH2LQG	离散时间H2综合
	DHINF	离散时间HOO综合
	H2LQG	连续时间H2综合
	HINF	连续时间HOO综合
	HINFOPT	HOO综合的R迭代
	LQG	LQG最优控制综合
	LTRU	LQG闭环传递补偿
	LTRY	LQG闭环传递补偿
	NORMH2	计算H2范数
	NORMHINF	计算HOO范数
	YOULA	YOULA参数化
演示示例	ACCDEMO	弹簧质量标准问题
	DINTDEMO	双积分器系统的HOO
	HINFDEMO	飞机或大型空间结构的H2或HOO设计示例
	LTRDEMO	LQR / LTR设计示例：飞机
	MRDEMO	鲁棒模型简化示例
	MUDEMO	U综合示例
	MUDEMO1	U综合示例
	RCTDEMO	鲁棒控制工具箱演示——主菜单

表 B.7　系统辨识工具箱

命令分类	函数指令	含　义
仿真和预测	IDSIM	仿真一给定的系统
	PE	计算预测误差
	POLY2TH	从给定的多项式中构造Θ矩阵
	PREDICT	M步超前预测

续表

命令分类	函数指令	含 义
数据处理	DTREND	从数据集中删除方位
	IDFILT	通过BUTTERWORTH滤波器对数据进行滤波
非参数化估计	COVF	估计数据矩阵的协方差矩阵
	CRA	相关分析
	ETFE	估计经验传递函数并计算周期图
	SPA	频谱分析
参数估计	AR	利用各种方法的AR信号模型
	ARMAX	ARMAX模型预测误差估计
	ARX	ARX模型的最小二乘估计
	BJ	BOX-JENKINS模型的预测误差估计
	CANSTART	具有初值参数估计的多变量模型
	IV4	ARX模型近似最优的IV估计
	IVAR	时间序列的AR部分的仪器IV估计
	IVX	单输出ARX模型的仪器可变估计
	OE	输出误差模型的预测误差估计
	PEM	一般线性模型的预测误差估计
建立模型结构	ARX2TH	ARX模型的Θ格式
	CANFORM	正则形模型结构
	MF2TH	将定义的模型结构封装入Θ模型格式中
	MODSTRUC	在MS2TH函数中使用的模型结构
	MS2TH	将标准状态空间参数封装入Θ格式中
	POLY2TH	从给定多项式中产生Θ矩阵
处理模型结构	FIXPAR	在状态空间和ARX模型结构中，找出要修正的参数
	SETT	在Θ结构中设置取样间隔
	THINIT	参数的(随机)初始值
	UNFIXPAR	在状态空间和ARX模型结构中，放松参数
模型变换	TH2ARX	变Θ格式模型为ARX模型
	TH2FF	求模型的频率响应及标准偏差
	TH2PAR	变Θ格式为参数和协方差阵
	TH2POLY	求给定模型相应的多项式
	TH2SS	变Θ格式为状态空间表示
	TH2TF	变Θ格式为传递函数表示
	TH2ZP	求零极点、静态增溢和标准偏差
	THC2THD	变连续时间模型为离散时间模型
	THD2THC	变离散时间模型为连续时间模型
模型表示	BODEPLOT	传递函数的BODE图或频谱
	FFPLOT	频域函数

续表

命令分类	函数指令	含　义
模型表示	IDPLOT	输入——输出数据
	NYQPLOT	传递函数灵敏的NYQUIST图
	PRESENT	屏幕上的参数模型
	ZPPLOT	零点和极点
信息提取	GETMFTH	获取定义模型结构的M文件的文件名
	GETNCAP	获取数据点数和参数个数
	GETFF	选取频率函数
	GETT	为某模型获取取样间隔
	GETZP	在由TH2ZP函数产生的零极点格式中，提取零点和极点
模型合法化	COMPARE	将仿真和预测的输出与测量输出比较
	IDSIM	仿真一给定的系统
	PE	预测误差
模型合法化	PREDICT	M步超前预测
	RESID	计算和测试与某模型相关的留数
估计模型的不确定性	IDSIMSD	在仿真模型响应中说明不确定性
	TH2FF	模型频率函数和标准偏差
	TH2ZP	零点、极点、静态增溢及其标准偏差
模型结构选择	ARXSTRUC	ARX模型类的损失函数
	YVSTRUC	单输出类的输出误差拟合
	SELSTRUC	根据各种准测选择模型结构
	STRUC	ARXSTRUC和IVSTRUC的典型结构矩阵
递归参数估计	RARX	对AR模型递归计算估值
	RARMAX	对ARMAX模型递归计算估值
	RBJ	对BOX-JENKINS模型递归计算估值
	ROE	对输出误差模型递归计算估值
	RPEM	对一般模型递归计算估值
	RPLR	对一般模型递归计算估值
	SEGMENT	分段数据并跟踪快变系统

表B.8　最优化工具箱

命令分类	函数指令	含　义
非线性最小化函数	ATTGOAL	达到多目标
	CONSTR	约束极小化
	FMIN	无约束极小化(标量情况)
	FMINS	利用单纯形搜索的无约束极小化

续表

命令分类	函数指令	含　义
非线性最小化函数	FMINU	利用梯度搜索的无约束极小化
	FSOLVE	非线性方程求解
	LEASTSQ	非线性最小二乘
	MINIMAX	极小极大求解
	SEMINF	半定极小化
矩阵问题极小化	LP	线性规划
	QP	二次规划
	NNLS	非负最小二乘
演示	OPTDEMO	演示菜单
	TUTDEMO	启动达到
	BANDEMO	香蕉型函数的极小化
	GOALDEMO	目标达到
	DFILDEMO	有限精度滤波器设计
	DATDEMO	数据推拟合成曲线
三次内插程序	CUBIC	内插4点以找出极大值
	CUBIC1	内插2点和梯度，以估计极小值
	CUBIC2	内插3点和1梯度
	CUBIC3	内插2点和梯度，以找出步长和极小值
二次内插程序	QUAD2	内插3点以找出极大值
	QUADINTER	内插3点以估计极小值
演示实用程序	EIGFUN	返回分类特征值的函数
	ELIMONE	消去一变量
	FILTFUN	频率响应和根
	FILTFUN2	频率响应范数和根
	FITFUN	返回拟合数据中的误差范数
	FITFUN2	返回拟合数据中的误差矢量
半定实用程序	SEMIFUN	半定问题转换成约束问题
	FINDMAX	在数据向量中内插极大值
半定实用程序	FINDMAX2	在数据阵中内插极大值
	V2SORT	分类两向量，然后删去丢失的元素
目标达到的实用程序	GOALFUN	目标达到问题转换成约束条件问题
	GOALGRA	变换目标达到问题中的梯度
测试程序	TOPTIM	最优化测试组
	TOPTIMF	最优化测试组的测试函数
	TOPTIMG	最优化测试组的测试函数梯度
其它	GRADERR	用于检查梯度的不一致性
	LSINT	初始化最小二乘程序的函数

续表

命令分类	函数指令	含 义
其它	OPTINT	初始化无约束极小化程序的函数
	SEARCHQ	线性搜索程序

表 B.9 神经网络工具箱

命令分类	函数指令	含 义
误差分析函数	ERRSURF	计算误差曲面
	PLOTEP	在误差曲面上绘制权和基位置图
	PLOTES	绘制误差曲面图
δ 函数	DELTALIN	对PURELIN神经元的 δ 函数
	DELTALOG	对LOGSIG神经元的 δ 函数
	DELTATAN	对TANSIG神经元的 δ 函数
设计	SOLVEHOP	设计HOPFIELD网络
	SOLVELIN	设计线性网络
	SOLVERB	设计径向基网络
	SOLVERBE	设计精确的径向基网络
初始化	INITC	竞争层初始化
	INITELM	ELMAN递归网络初始化
	INITFF	至多三层的前向网络初始化
	INITLIN	线性层初始化
	INITLVQ	LVQ网络初始化
	INITP	感知层初始化
	INITSM	自组织映射初始化
	MIDPOINT	产生终点值
	NWLOG	对LOGSIG神经元产生NGUYER-WIDROW随机数
	NWTAN	对TANSIG神经元产生NGUYEN-WIDROW随机数
	RANDNC	产生归一化列随机数
	RANDNR	产生归一化行随机数
	RANDS	产生对称随机数
学习规则	LEARNBP	反向演播学习规则
	LEARNBPM	带预测的反向演播学习规则
	LEARNH	HEBB学习规则
	LEARNHD	退化的HEBB学习规则
	LEARNIS	内星学习规则
	LEARNK	KOHONEN学习规则
	LEARNLM	LE-+VENBERG-MARQUARDT学习规则
	LEARNLVQ	学习矢量量化规则

续表

命令分类	函数指令	含 义
学习规则	LEARNOS	外星学习规则
	LEARNP	感知层学习规则
	LEARNPN	归一化的感知层学习规则
	LEARNWH	WIDROW-HOFF学习规则
矩阵	COMBVEC	创建所有的矢量集
	DELAYSIG	从信号矩阵中建立退化的信号矩阵
	DIST	计算矢量距离
矩阵	IND2VEC	变下标矢量为稀疏矩阵表示
	NORMC	归一化矩阵列
	NORMR	归一化矩阵行
	PNORMC	伪归一化矩阵列
	QUANT	离散化成某数值的整数倍
	SNMSQR	平方和
	VECT2IND	变稀疏矩阵表示为下标矢量
邻域	NBDIST	使用矢量距离的邻域阵
	NBGRID	使用栅格距离的邻域阵
	NBMAN	使用MANHATTAN距离的邻域阵
绘图	BARERR	每个输出矢量的误差条形图表
	HINTONW	绘制权值图
	HINTONWB	绘制权值和偏差图
	PLOTERR	绘出网络误差与时间的关系
	PLOTES	绘制误差曲面
	PLOTFA	绘出目标模式及网络函数的逼近
	PLOTPV	绘出限幅神经元的感知器分类
	PLOTSM	绘制自组织映射图
	PLOTTR	绘出网络误差记录及自适应学习速率
	PLOTVEC	用不同颜色绘制矢量
仿真	SIMUC	竞争层仿真
	SIMUELM	ELMAN递归网络仿真
	SIMUFF	前向网络仿真
	SIMUHOP	HOPFIELD网络仿真
	SIMULIN	线性层仿真
	SIMUP	感知层仿真
	SIMURB	径向基网络仿真
	SIMUSM	自组织映射仿真
训练	TRAINBP	利用反向演播训练前向网络
	TRAINC	训练竞争层网络

续表

命令分类	函数指令	含 义
训练	TRAINELM	训练ELMAN递归网络
	TRAINLVQ	训练LVQ网络
	TRAINP	利用感知规则训练感知层
	TRAINPN	利用归一化感知规则训练感知层
	TRAINSM	利用KOHONEN规则训练自组织映射
	TRAINWH	利用WIDROW
	TRAIRBPX	利用快速反应演播训练网络
传递函数	COMPET	竞争层传递函数
	HARDLIM	硬限幅传递函数
	HARDLIMS	对称硬限幅传递函数
	LOGSIG	线性S型传递函数
	PURELIN	线性传递函数
	RADBAS	径向基传递函数
	SATLINS	对称饱和线性传递函数
	TANSIG	正切S型传递函数

表 B.10 模糊系统工具箱

命令分类	函数指令	含 义
GUI编辑器	FUZZY	基本FIS(模糊推理系统)编辑器
	MFEDIT	隶属度函数编辑器
	RELEVIEW	规则观察器及模糊推理框图
	RULEEDIT	规则编辑器及(句法)分析程序
	SURFVIEW	输出曲面观测器
隶属度函数	DSIGMF	两个“S”形隶属度函数的差
	GAUSS2MF	双边高斯曲线录属度函数
	GAUSSMF	高斯曲线录属度函数
	GBELLMF	广义钟形隶属度函数
	PIMF	π 形隶属度函数
	PSIGMF	两个“S”形隶属度函数
	SIGMF	“SIGMOID(S)”形隶属度函数
	SMF	“S”形隶属度函数
	TRAPMF	梯形隶属度函数
	TRIMF	三角形隶属度函数
	ZMF	“Z”形隶属度函数
命令行FIS函数	ADDMF	将隶属度函数加到FIS中
	ADDRULE	将规则加到FIS中

续表

命令分类	函数指令	含 义
命令行FIS函数	ADDVAR	将变量加FISK
	DEFUZZ	去模糊隶属度函数
	EVALFIS	完成模糊推理计算
	EVALMF	隶属度函数计算
	GENSURF	产生FIS输出曲面
	GETFIS	获得模糊系统的特性
	MF2MF	在函数之间变换参数
	NEWFIS	产生新的FIS
	PARSRULE	分析模糊规则
	PLOTFIS	显示FIS输入/输出图
	PLOTMF	显示出一个变量的所有隶属度函数
	READFIS	从磁盘中装入FIS
	RMMF	从FIS删除隶属度函数灵敏
	RMVAR	从FIS中删除变量
	SETFIS	设置模糊系统性
	SHOWFIS	显示带注释的FIS
	SHOWRULE	显示FIS规则
	WRITEFIS	在磁盘中保存FIS
先进技术	ANFIS	SUGENO-TYPEFIS的训练程序
	FCM	利用模糊C平均聚集方法找出簇
	GENFIS1	利用一般方法产生FIS矩阵
	GENFIS2	利用减法聚集方法产生FIS矩阵
	SUBCLUST	利用减法聚集方法估计簇中心

附录C 习题参考答案

第1章

(1)，(2)，(3)略。

(4) 在MatLab的工作空间中输入：

```
A=[  2  4  7
     3  1  6]
B=[  1  3  5
     2  2  5
     4  8  0]
save mat1
quit
```

(5) 现打开MatLab，查看变量：

who

发现没有任何变量，然后装入mat1.mat文件：

load mat1

查看当前的工作空间中的变量：

who

Your variables are:

A　　　　B

(6) 用path命令查看当前的搜索路径：

```
path

    MatLabPATH

E:\MatLab_BOOK\LAST_M_FILE\image
F:\MatLab\toolbox\MatLab\general
F:\MatLab\toolbox\MatLab\ops
F:\MatLab\toolbox\MatLab\lang
F:\MatLab\toolbox\MatLab\elmat
F:\MatLab\toolbox\MatLab\elfun
F:\MatLab\toolbox\MatLab\specfun
F:\MatLab\toolbox\MatLab\matfun
F:\MatLab\toolbox\MatLab\datafun
F:\MatLab\toolbox\MatLab\polyfun
F:\MatLab\toolbox\MatLab\funfun
```

```
F:\MatLab\toolbox\MatLab\sparfun
F:\MatLab\toolbox\MatLab\graph2d
F:\MatLab\toolbox\MatLab\graph3d
F:\MatLab\toolbox\MatLab\specgraph
F:\MatLab\toolbox\MatLab\graphics
F:\MatLab\toolbox\MatLab\uitools
F:\MatLab\toolbox\MatLab\strfun
F:\MatLab\toolbox\MatLab\iofun
F:\MatLab\toolbox\MatLab\timefun
F:\MatLab\toolbox\MatLab\datatypes
F:\MatLab\toolbox\MatLab\winfun
F:\MatLab\toolbox\MatLab\demos
F:\MatLab\toolbox\compiler
F:\MatLab\toolbox\symbolic
F:\MatLab\toolbox\dspblks\dspblks
F:\MatLab\toolbox\dspblks\dspmex
F:\MatLab\toolbox\dspblks\dspdemos
F:\MatLab\toolbox\fdident\fdident
F:\MatLab\toolbox\fdident\fddemos
F:\MatLab\toolbox\stats
F:\MatLab\toolbox\signal
F:\MatLab\toolbox\optim
F:\MatLab\toolbox\tour
F:\MatLab\toolbox\local
```

用cd命令查看当前的路径：

```
cd
F:\MatLab\bin
```

建立名为dirname的路径于F盘，然后将其加入搜索路径中：

```
path('f:\dirname',path)
或path(path,'f:\dirname')
```

将其设为当前路径：

```
cd f:\dirname
cd
f:\dirname
```

(7)，(8)，(9)略。

(10) 在工作空间中输入：

```
A=eye(3,3)
A =
     1     0     0
     0     1     0
     0     0     1
B=['how are you' ]
```

```
B =
how are you
```

查看变量类型：

```
who
Your variables are:
A        B
clear A
save data
clear all
who
load data
who
Your variables are:
B
B
B =
how are you
```

第2章

(1)
```
a =
     2     3     4
     4     2     8
     0     4     1
b =
     9     0     1
     0     1     3
     9     3     0
a+b
ans =
    11     3     5
     4     3    11
     9     7     1
a-b
ans =
    -7     3     3
     4     1     5
    -9     1     1
a*b
ans =
    54    15    11
   108    26    10
     9     7    12
a.*b
ans =
```

```
    18     0     4
     0     2    24
     0    12     0
a/b
ans =
   -0.3000    1.4333    0.5222
    0.6000    2.4667   -0.1556
   -1.1000    0.7000    1.1000
a./b
Warning: Divide by zero.
ans =
    0.2222       Inf    4.0000
       Inf    2.0000    2.6667
         0    1.3333       Inf
```

(2) pi

```
ans =

   3.1416

format short
pi

ans =

   3.1416

format long
pi

ans =

  3.14159265358979

format short e
pi

ans =

  3.1416e+000

format long e
pi

ans =

   3.141592653589793e+000
```

```
format short g
pi

ans =

      3.1416

format long g
pi

ans =

         3.14159265358979

format hex
pi

ans =

   400921fb54442d18

format bank
pi

ans =

         3.14

format rat
pi

ans =

   355/113

format +
pi

ans =

+

format compact
pi
ans =
+
```

```
format loose
pi

ans =

+
```

(3)
```
pai=0;
flag=1;
for i=1:11
   pai=pai+flag/(2*i-1);
   flag=0-flag;
end
pai=pai*4
```

执行结果为 3.2323

(4)
```
answ=0;

flag=1;
for i=1:30
   flag=flag*i;
   answ=answ+flag;
end
answ
```

执行结果为:

```
answ =

  2.7441e+032
```

(5) function y=www(B)

```
n=length(B);
y=0;
flag=0;
for i=1:n
   if (B(i)>='a' & B(i)<='z')|(B(i)>='A' & B(i)<='Z')
      if flag==0
         y=y+1;
         flag=1;
      end
   else
      if flag==1
         flag=0;
      end
   end
end

B =['how are you']
```

```
www(B)
ans =
     3
```

(6) function y=t26(B)

```
n=length(B);
y=zeros(4,1);
for i=1:n
   if (B(i)>='a' & B(i)<='z')|(B(i)>='A' & B(i)<='Z')
      y(1)=y(1)+1;
   elseif B(i)>='0' & B(i)<='9'
      y(2)=y(2)+1;
   elseif B(i)==' '
      y(3)=y(3)+1;
   else
      y(4)=y(4)+1;
   end
end
```

(7) function y=lerond(n,x)

```
if n==0
   y=1;
elseif n==1
   y=x;
else
   y=((2*n-1)*x*lerond(n-1,x)-(n-1)*lerond(n-2,x))/n;
end
```

(8) A(find(A==-1))=-2

```
A =

     1     4     6     8
     2     3     7     0
    -2    -2     5     7
     1     1    -2     0

B(find(B<0))=1

B =

     1     1     6     1
     1     1     3     1
     0     0     1     3
     2     6     1     0

B(find(B==0))=A(find(B==0))
```

```
B =

     1     1     6     1
     1     1     3     1
    -2    -2     1     3
     2     6     1     0
```

(9)

```
① A>B

ans =

     1     1     0     1
     1     1     1     1
     0     0     1     1
     0     0     0     0
② A==0

ans =

     0     0     0     0
     0     0     0     1
     0     0     0     0
     0     0     0     1
③ A>1&A<7

ans =

     0     1     1     0
     1     1     0     0
     0     0     1     0
     0     0     0     0
```

(10) s1=deblank(s1)

```
s1 =

this is an example of MatLab
s1=lower(s1)

s1 =

this is an example of MatLab
sum(s1==' ')

ans =
```

```
    5
strrep(s1,'MatLab','MatLab')

ans =

this is an example of MatLab
```

第4章

(1) AB

```
ans =
    22     7    12
    16     8     7
    14     5     7
A*B-B*A
ans =
    18     4     2
    13     0    -6
     7    -9   -18
A'*B
ans =
     8     3     3
    11     8     2
    22    13     5
```

(2)

```
C =
     1     0     2    -5
    -1     2     1     3
     2    -1     0     1
     1     3     4     2
rank(C)
ans =
       3
det(C)
ans =
    0
trace(C)
ans =
       5
norm(C)
ans =
   6.7156
cond(C)
ans =
```

```
7.1880e+015
D =

     1     2     0     0
    -1     3     0     0
     0     0     2    -1
     0     0     5     4
det(D)
ans =
    65
rank(D)
ans =
     4
trace(D_
ans =
    10
norm(D)
ans =
    6.4787
cond(D)
ans =
4.6880
E =

     0     0     1    -1     2
     0     0     2     0    -3
     0     0    -1     4     0
    -1     2     4     0    -1
     3    -2     1     5     1
det(E)
ans =
  -100
rank(E)
ans =
     5
trace(E)
ans =
     0
norm(E)
ans =
    7.2450
cond(E)
ans =
   22.8735
```

(3)

① 1.1818

```
    1.3766
   -1.5974
② 1
   -1
  0
  2
③
 -11.3333
 -2.6667
 -8.6667
 -9.0000
```

或：

```
 -20.0000
   6.0000
        0
  -9.0000
④
        0
   0.5000
   1.0000
```

(4)

① 只有 x 和 y 共线。

② 只有 x 和 z 共线。

提示：用书中所提供的iscline函数来判断

(5)

① 不共面。

② 共面。

提示：用书中所给出的iscface函数来判断

(6)

① 0.6667

② 2.2946

提示：用 ptof 函数来求解

(7)

① 9.6923

② 2.0909

提示：用 ltol 函数来求解

(8) 垂足为：

1.1493 −4.9851 1.8507

对称点为：

0.2985 −5.9701 −1.2985

所用到的函数为prorthl

(9)

```
v1=p1-p2
v1 =
    -4     5     1
v2=p1-p3
v2 =
     0    -4     4
c=cross_product(v2,v1)
c =
   -24   -16   -16
area=0.5*radvec(c)
area =
   16.4924                    %面积
l=area*2/radvec(p1-p2)
l =
5.897                         %高
```

(10)

① x×y

```
cross_product(x,y)
ans =
     2    -2     2
```

② y×z

```
cross_product(y,z)
ans =
     4    -3     2
```

③ x×y×z

```
mixed_product(x,y,z)
ans =
     2
```

第5章

(1)

①

```
v =
         0         0    0.2722
         0    0.7071    0.1361
    1.0000   -0.7071   -0.9526
d =
     2     0     0
     0    -1     0
     0     0     1
```

②

```
v =
    1.0000         0    0.9998
         0   -0.7071    0.0158
```

```
        0    0.7071   -0.0158
d =
     2     0     0
     0     2     0
     0     0     2
```

③
```
v =
   -0.8294    0.1233   -0.2166   -0.5000
   -0.4901   -0.6063    0.3772    0.5000
   -0.2545    0.7838    0.2664    0.5000
   -0.0848   -0.0542   -0.8602    0.5000
d =
    2.0000         0         0         0
         0    2.0000         0         0
         0         0    2.0000         0
         0         0         0   -2.0000
```

(2)

① [1.0000   -4.0000   2.0000   -7.0000]

② [1.0000   -5.0000   8.0000   -4.0000]

(3)

① 可以对角化，对角化后矩阵为：

```
0         0         0
0    3.0000         0
0         0    2.0000
```

可逆矩阵为：

```
 0.3015   -0.7071    0.0000
-0.9045    0.0000    0.8944
-0.3015    0.7071    0.4472
```

② 可以对角化，对角化后的矩阵为：

```
-1     0     0
 0     4     0
 0     0     1
```

可逆矩阵为：

```
-0.8321   -0.7071   -0.1374
      0         0    0.8242
 0.5547   -0.7071   -0.5494
```

③ 可以对角化，对角化后的矩阵为：

```
2     0
0    -3
```

可逆矩阵为：

```
0.9701   -0.7071
0.2425    0.7071
```

④ 不可以对角化。

(4)

① 可逆矩阵为：

```
-0.6133    0.5390    0.5774
 0.7734    0.2616    0.5774
-0.1601   -0.8006    0.5774
```

对角矩阵为：

```
1.0000         0         0
     0    1.0000         0
     0         0   -2.0000
```

② 可逆矩阵为：

```
      0    0.7071    0.7071
 1.0000         0         0
      0   -0.7071    0.7071
```

对角矩阵为：

```
2    0    0
0    2    0
0    0    4
```

③ 可逆矩阵为：

```
-0.5000   -0.5000    0.5000    0.5000
 0.5000   -0.5000   -0.5000    0.5000
 0.5000    0.5000    0.5000    0.5000
-0.5000    0.5000   -0.5000    0.5000
```

对角矩阵为：

```
-3.0000         0         0         0
      0   -5.0000         0         0
      0         0    3.0000         0
      0         0         0    5.0000
```

(5) 求解结果为：

```
-1    0    0
 0   -1    1
 0    0   -1
```

(6) 最大公因式为：x+1

最小公倍式为：$x^8-2x^7+5x^6-4x^5+4x^4+3x^3-4x^2+6x-3$

(7)

① 2.0000 + 1.7321i

```
  2.0000 - 1.7321i
  2.0000
```

②
```
  7.1589
 -1.2754
 -0.4418 + 0.3653i
 -0.4418 - 0.3653i
```

(8)

① **LLT 分解：**

```
    1.4142    0.7071    0.7071
         0    1.2247    0.4082
         0         0    1.1547
```

LDL分解：

```
l =
    1.0000         0         0
    0.5000    1.0000         0
    0.5000    0.3333    1.0000
d =
    2.0000         0         0
         0    1.5000         0
         0         0    1.3333
```

LU分解：

```
l =
    1.0000         0         0
    0.5000    1.0000         0
    0.5000    0.3333    1.0000
u =
    2.0000    1.0000    1.0000
         0    1.5000    0.5000
         0         0    1.3333
```

QR分解：

```
q =
   -0.8165    0.4924   -0.3015
   -0.4082   -0.8616   -0.3015
   -0.4082   -0.1231    0.9045
r =
   -2.4495   -2.0412   -2.0412
         0   -1.3540   -0.6155
         0         0    1.2060
e =
     1     0     0
     0     1     0
     0     0     1
```

② LLT 分解:

```
1.7321   -0.5774        0
     0    1.6330   0.6124
     0         0   0.7906
```

LDL分解:

```
l =
    1.0000         0         0
   -0.3333    1.0000         0
         0    0.3750    1.0000
d =
    3.0000         0         0
         0    2.6667         0
         0         0    0.6250
```

LU分解:

```
l =
    1.0000         0         0
   -0.3333    1.0000         0
         0    0.3750    1.0000
u =
    3.0000   -1.0000         0
         0    2.6667    1.0000
         0         0    0.6250
```

QR分解:

```
q =
   -0.3015   -0.9463   -0.1162
    0.9045   -0.2453   -0.3487
    0.3015   -0.2103    0.9300
r =
    3.3166   -1.8091    1.2060
         0   -2.5937   -0.4556
         0         0    0.5812
e =
     0     1     0
     1     0     0
     0     0     1
```

(9)

① P =

```
-0.6133    0.5390    0.5774
 0.7734    0.2616    0.5774
-0.1601   -0.8006    0.5774
```

二次型为：$-y_1^2-y_2^2+2y_3^2$

② P =

```
0.4927   -0.5392    0.6831
-0.8433   -0.1019    0.5277
0.2149    0.8360    0.5049
```

二次型为：$1.3218y_1^2-2.8237y_2^2+7.5020y_3^2$

(10)

```
X1=1.1708*sin(0.6180t)+0.1708*sin(1.6180t)
X2=1.8944*sin(0.6180t)-0.1056*sin(1.6180t)
X3=1.8944*sin(0.6180t)+-0.1056*sin(1.6180t)
X4=1.1708*sin(0.6180t)+0.1708*sin(1.6180t)
```

第6章

- 极限、导数与微分

(1) 函数 $y=(1+x)^{1/x}$ 在 x=0 处的极限。

```
syms x
y=(1+x)^(1/x);
limit(y,x,0)
ans =
exp(1)
```

(2) 函数 y=sin3x/tg5x 在 x=0 处的极限。

```
syms x
y=sin(3*x)/tan(5*x);
limit(y,x,0)
 ans =
 3/5
```

(3) 函数 $y=xn/3^x$ 在趋向正无穷处的极限值。

提示：用取绝对值函数 abs

```
syms n x
y=abs(x)*n/3^(abs(x));
limit(y,x,inf)
ans =
0
```

(4) 函数 $y=\ln^2x/x^3$ 在趋向正无穷处的极限值。

```
syms x
y=(log(abs(x)))^2/abs(x)^3;
limit(y,x,inf)
ans =
0
```

(5) 求函数 $y=1/(x^2-3x+3)$的 50 阶导数。

```
syms x
y =1/(x^2-3*x+3);
diff(y,x,50)
ans =
304140932017133780436126081660647688443776415689605120000000000000/(x
^2-3*x+3)^51*(2*x-3)^50-1065538183853052897199196662384724313784019315356
181911830528000000000000000/(x^2-3*x+3)^42*(2*x-3)^32+913791886456876469
0210114665385384709008589066040421174476800000000000000/(x^2-3*x+3)^31*(2*
x-3)^10+8234540149259472207220999674908221860451714407481231443335577600
0000000000/(x^2-3*x+3)^39*(2*x-3)^26-37158692096032135045186096540363125
918537982985651626180608000000000000000/(x^2-3*x+3)^46*(2*x-3)^40-980067206
538636889402469657380602042624873380969322400186368000000000000000/(x^2-3*x
+3)^44*(2*x-3)^36-429205280002472280903808416101434736332221607647353115
770880000000000000000/(x^2-3*x+3)^32*(2*x-3)^12-354872585355890268298929068
43375768749050938197128404868792320000000000000000/(x^2-3*x+3)^34*(2*x-3)^
16+34307097131532690433195022011321059256457979689787457536000000000000/
(x^2-3*x+3)^49*(2*x-3)^46+988458029055684786417409765397104987442273350 9
912166400000000000000000/(x^2-3*x+3)^27*(2*x-3)^2-98783245982107466497245752
38273021008507746539083696326049792000000000000000/(x^2-3*x+3)^36*(2*x-3)
^20-304140932017133780436126081660647688443776415689605120000000000000/(x
^2-3*x+3)^26+25780826302005572537112270465503573640823101483739816013070
336000000000000/(x^2-3*x+3)^41*(2*x-3)^30-1305416980652680670030016380 76
934067271551272372006016778240000000000000000/(x^2-3*x+3)^30*(2*x-3)^8-1490
2905668839555241370178001371736733745044368790650880000000000000000/(x^2-3*x
+3)^50*(2*x-3)^48-493164521265782424977178441412740226811583458040694702
080000000000000/(x^2-3*x+3)^48*(2*x-3)^44+143382862770056674060173360983
3364393901048007964784035102720000000000000000/(x^2-3*x+3)^33*(2*x-3)^14+3
58978104720547232746253421017313538868412924424809948905472000000000000000
/(x^2-3*x+3)^43*(2*x-3)^34-108349212490256213252907890459318708690154136
940542518991257600000000000000000/(x^2-3*x+3)^38*(2*x-3)^24+214694665443741
224705519668899875838640441679472653840154624000000000000000/(x^2-3*x+3)^45
*(2*x-3)^38+4963123799121597596046923463579279303869765439430821150720000
000000000/(x^2-3*x+3)^47*(2*x-3)^42-509757247335110184256538075113366115
170820415701219089349345280000000000000000/(x^2-3*x+3)^40*(2*x-3)^28+115460
936862203532269508022265528816982558076430848398616166400000000000000000/(x
^2-3*x+3)^37*(2*x-3)^22+1145820547281349804415061400048324101443083268469
018329088000000000000000/(x^2-3*x+3)^29*(2*x-3)^6-53376733569006978466540
12733144366932188276095352569856000000000000000/(x^2-3*x+3)^28*(2*x-3)^4+
670314883450014951231310462597097854148739943723536536410521600000000000
00/(x^2-3*x+3)^35*(2*x-3)^18
```

(6) 函数 $y=a\sin(b\exp(c^t))+t^a$ 在 $t=b$ 处的 3 阶导数。

```
syms a b c t
y=a*sin(b*exp(c^t))+t^a;
diff(y,t,b)
```

```
ans =
a*cos(b*exp(c^t))*exp(c^t)
```

- 积分

(1) 求下列不定积分:

① 1/sinx

```
int('1/sin(x)')
ans =
log(csc(x)-cot(x))
```

② $1/\sin^2 x$

```
y=1/(sin(x))^2
int(y)
ans =
-1/sin(x)*cos(x)
```

③ $1/\sin^3 x$

```
y=1/(sin(x))^3;
int(y)
ans =
-1/2/sin(x)^2*cos(x)+1/2*log(csc(x)-cot(x))
```

④ $1/(a^2-x^2)$

```
syms a
y=1/(a^2-x^2);
int(y)
ans =
-1/2/a*log(a-x)+1/2/a*log(a+x)
```

⑤ $(2-\sin x)/(\sin^2 x)$

```
y =(2-sin(x))/sin(x)^2;
int(y)
ans =
-2/sin(x)*cos(x)-log(csc(x)-cot(x))
```

⑥ $((x^2-3)^{1/2}-(x^2+3)^{1/2})/(x^4-9)^{1/2}$

```
y=((x^2-3)^0.5-(x^2+3)^0.5)/(x^4-9)^0.5;
int(y)
ans =
asinh(1/3*3^(1/2)*x)*(1/(x^2+3))^(1/2)*(x^2+3)^(1/2)-log(x+(x^2-3)^(
1/2))*(1/(x^2-3))^(1/2)*(x^2-3)^(1/2)
```

⑦ $(ac+b)/(ac^2+bc+c)$关于 c 的不定积分

```
syms a b c
y=(a*c+b)/(a*c^2+b*c+c);
```

```
int(y,c)
ans =
b/(b+1)*log(c)+1/(b+1)*log(a*c+b+1)
```

(2) 求下列定积分及广义积分:

① $y=(x^2+a)^{1/2}$ 在[-2,2]

```
  syms a
y=(x^2+a)^0.5;
int(y,a,-2,2)
ans =
2/3*(x^2+2)^(3/2)-2/3*(x^2-2)^(3/2)
```

② $(\sin x\cos x)^2$ 在[π, π]

```
y=(sin(x)*cos(x))^2;
int(y,-pi,pi)
ans =
1/4*pi
```

③ besselj(0,x) 在[1,2]

```
y=besselj(0,x);
int(y,1,2)
ans =
2*besselj(0,2)-pi*StruveH(1,2)*besselj(0,2)+pi*StruveH(0,2)*besselj(
1,2)-besselj(0,1)+1/2*pi*StruveH(1,1)*besselj(0,1)-1/2*pi*StruveH(0,1)*b
esselj(1,1)
```

- 化简、提取与替换带入

(1) 对下式分别用 simple 和 simplify 命令环境。

① cos(3acos(x))

```
simplify:
simplify(y)
ans =
cos(3*acos(x))

simple:
y=cos(3*acos(x));

simple(y)
simplify:
cos(3*acos(x))

radsimp:
cos(3*acos(x))
combine(trig):
cos(3*acos(x))
```

```
factor:
cos(3*acos(x))

expand:
4*x^3-3*x

convert(exp):
1/2*exp(3*i*acos(x))+1/2/exp(3*i*acos(x))

convert(sincos):
cos(3*acos(x))

convert(tan):
 (1-tan(3/2*acos(x))^2)/(1+tan(3/2*acos(x))^2)

collect(x):
cos(3*acos(x))

ans =
4*x^3-3*x
```

② x^2-3x+sin(acos3x)

```
simplify:
y=x^2-3*x+sin(acos(3*x));
simplify(y)
ans =
x^2-3*x-(1-9*x^2)^(1/2)

simple:
simple(y)
simplify:
x^2-3*x+(1-9*x^2)^(1/2)

radsimp:
x^2-3*x+(1-9*x^2)^(1/2)

combine(trig):
x^2-3*x+(1-9*x^2)^(1/2)

factor:
x^2-3*x+(-(3*x-1)*(3*x+1))^(1/2)

expand:
x^2-3*x+(1-9*x^2)^(1/2)

convert(exp):
x^2-3*x+(1-9*x^2)^(1/2)
```

```
convert(sincos):
x^2-3*x+(1-9*x^2)^(1/2)

convert(tan):
x^2-3*x+(1-9*x^2)^(1/2)

collect(x):
x^2-3*x+(1-9*x^2)^(1/2)

ans =
x^2-3*x+(1-9*x^2)^(1/2)
```

③ sin2x/tgx

```
simplify:
y=sin(2*x)/tan(x);
simplify(y)
ans =
2*cos(x)^2

simple:
simple(y)
simplify:
2*cos(x)^2

radsimp:
sin(2*x)/tan(x)

combine(trig):
sin(2*x)/tan(x)

factor:
sin(2*x)/tan(x)

expand:
2/tan(x)*sin(x)*cos(x)

convert(exp):
1/2*(exp(2*i*x)-1/exp(2*i*x))/(exp(i*x)^2-1)*(exp(i*x)^2+1)

convert(sincos):
sin(2*x)/sin(x)*cos(x)

convert(tan):
2/(1+tan(x)^2)

collect(x):
sin(2*x)/tan(x)
```

```
ans =
2*cos(x)^2
```

(2) 对方程解进行提取与替换代入。

方程解为:

```
t=solve('a*x^6+b*x^2+c')


t=solve('a*x^6+b*x^2+c')
t =
[1/6/a/(-12*(9*c-3^(1/2)*((4*b^3+27*c^2*a)/a)^(1/2))*a^2)^(1/3)*(-6*
a*(-12*(9*c-3^(1/2)*((4*b^3+27*c^2*a)/a)^(1/2))*a^2)^(1/3)*(-(-12*(9*c-3
^(1/2)*((4*b^3+27*c^2*a)/a)^(1/2))*a^2)^(2/3)+12*b*a))^(1/2)]
[-1/6/a/(-12*(9*c-3^(1/2)*((4*b^3+27*c^2*a)/a)^(1/2))*a^2)^(1/3)*(-6
*a*(-12*(9*c-3^(1/2)*((4*b^3+27*c^2*a)/a)^(1/2))*a^2)^(1/3)*(-(-12*(9*c-
3^(1/2)*((4*b^3+27*c^2*a)/a)^(1/2))*a^2)^(2/3)+12*b*a))^(1/2)]
[1/6/a/(-12*(9*c-3^(1/2)*((4*b^3+27*c^2*a)/a)^(1/2))*a^2)^(1/3)*3^(1
/2)*(a*(-12*(9*c-3^(1/2)*((4*b^3+27*c^2*a)/a)^(1/2))*a^2)^(1/3)*(12*i*3^
(1/2)*b*a-(-12*(9*c-3^(1/2)*((4*b^3+27*c^2*a)/a)^(1/2))*a^2)^(2/3)+12*b*
a+i*3^(1/2)*(-12*(9*c-3^(1/2)*((4*b^3+27*c^2*a)/a)^(1/2))*a^2)^(2/3)))^(
1/2)]
[-1/6/a/(-12*(9*c-3^(1/2)*((4*b^3+27*c^2*a)/a)^(1/2))*a^2)^(1/3)*3^(
1/2)*(a*(-12*(9*c-3^(1/2)*((4*b^3+27*c^2*a)/a)^(1/2))*a^2)^(1/3)*(12*i*3
^(1/2)*b*a-(-12*(9*c-3^(1/2)*((4*b^3+27*c^2*a)/a)^(1/2))*a^2)^(2/3)+12*b
*a+i*3^(1/2)*(-12*(9*c-3^(1/2)*((4*b^3+27*c^2*a)/a)^(1/2))*a^2)^(2/3)))^
(1/2)]
[1/6/a/(-12*(9*c-3^(1/2)*((4*b^3+27*c^2*a)/a)^(1/2))*a^2)^(1/3)*(-3*
a*(-12*(9*c-3^(1/2)*((4*b^3+27*c^2*a)/a)^(1/2))*a^2)^(1/3)*(12*i*3^(1/2)
*b*a+(-12*(9*c-3^(1/2)*((4*b^3+27*c^2*a)/a)^(1/2))*a^2)^(2/3)-12*b*a+i*3
^(1/2)*(-12*(9*c-3^(1/2)*((4*b^3+27*c^2*a)/a)^(1/2))*a^2)^(2/3)))^(1/2)]
[-1/6/a/(-12*(9*c-3^(1/2)*((4*b^3+27*c^2*a)/a)^(1/2))*a^2)^(1/3)*(-3
*a*(-12*(9*c-3^(1/2)*((4*b^3+27*c^2*a)/a)^(1/2))*a^2)^(1/3)*(12*i*3^(1/2
)*b*a+(-12*(9*c-3^(1/2)*((4*b^3+27*c^2*a)/a)^(1/2))*a^2)^(2/3)-12*b*a+i*
3^(1/2)*(-12*(9*c-3^(1/2)*((4*b^3+27*c^2*a)/a)^(1/2))*a^2)^(2/3)))^(1/2)
]

[r,s]=subexpr(t,'s')
r =
[1/6/a/s^(1/3)*(-6*a*s^(1/3)*(-s^(2/3)+12*b*a))^(1/2)]
[-1/6/a/s^(1/3)*(-6*a*s^(1/3)*(-s^(2/3)+12*b*a))^(1/2)]
[1/6/a/s^(1/3)*3^(1/2)*(a*s^(1/3)*(12*i*3^(1/2)*b*a-s^(2/3)+12*b*a+i
*3^(1/2)*s^(2/3)))^(1/2)]
[-1/6/a/s^(1/3)*3^(1/2)*(a*s^(1/3)*(12*i*3^(1/2)*b*a-s^(2/3)+12*b*a-
i*3^(1/2)*s^(2/3)))^(1/2)]
[1/6/a/s^(1/3)*(-3*a*s^(1/3)*(12*i*3^(1/2)*b*a+s^(2/3)-12*b*a+i*3^(1
```

```
/2)*s^(2/3)))^(1/2)]
    [-1/6/a/s^(1/3)*(-3*a*s^(1/3)*(12*i*3^(1/2)*b*a+s^(2/3)-12*b*a+i*3^(
1/2)*s^(2/3)))^(1/2)]

    s =
    -12*(9*c-3^(1/2)*((4*b^3+27*c^2*a)/a)^(1/2))*a^2
```

- 级数求和

(1) $(z-1)^n/(n^2 2^n)$　　　　$n=1\rightarrow\infty$

```
syms z n
y=(z-1)^n/(n^2*2^n);
symsum(y,n,1,inf)
ans =
(1/2*z-1/2)*hypergeom([1, 1, 1],[2, 2],1/2*z-1/2)
```

(2) $(-1)^{n+1}nz^n$　　　　$z\in C, n=1\rightarrow\infty$

```
y=(-1)^(n+1)*n*z^n;
symsum(y,n,1,inf)
ans =
z/(z+1)^2
```

(3) $(3n+1)(z-1)^n$　　　　$z\in C, n=1\rightarrow\infty$

```
y=(3*n+1)*(z-1)^n;
symsum(y,n,1,inf)
ans =
(4*z-4)*(-1/(z-2)+3/4/(z-2)^2*(z-1))
```

- 泰勒、傅里叶级数展开

(1) 一元函数 taylor 展开

① $y=e^{-2x}$　　　　x=0 处 6 阶麦克劳林型泰勒展开式

```
y=exp(-2*x);
taylor(y,6)
ans =
1-2*x+2*x^2-4/3*x^3+2/3*x^4-4/15*x^5
```

② $y= a\sin(bx)x/b$　　　　b=3 处变量 b 的 4 阶麦克劳林型泰勒展开式

```
y=a*sin(b*x)*x/b;
taylor(y,b,4,3)
ans =
1/3*a*sin(3*x)*x+(1/3*a*cos(3*x)*x^2-1/9*a*sin(3*x)*x)*(b-3)+(-1/6*a
*sin(3*x)*x^3-1/9*a*cos(3*x)*x^2+1/27*a*sin(3*x)*x)*(b-3)^2+(-1/18*a*cos
(3*x)*x^4+1/18*a*sin(3*x)*x^3+1/27*a*cos(3*x)*x^2-1/81*a*sin(3*x)*x)*(b-
3)^3
```

③ $y= x/\sin x$　　　　x=2 处 10 阶麦克劳林型泰勒展开式

```
syms x
```

```
y= x/sin(x);
taylor(y,10,2)

ans =
2/sin(2)+(1-2/sin(2)*cos(2))/sin(2)*(x-2)+(1-(sin(2)-2*cos(2))/sin(2
)^2*cos(2))/sin(2)*(x-2)^2+(1/2*(sin(2)-2*cos(2))/sin(2)+1/3/sin(2)*cos(
2)-(sin(2)^2-cos(2)*sin(2)+2*cos(2)^2)/sin(2)^3*cos(2))/sin(2)*(x-2)^3+(
1/6*(sin(2)-2*cos(2))/sin(2)^2*cos(2)-1/12+1/2*(sin(2)^2-cos(2)*sin(2)+2
*cos(2)^2)/sin(2)^2-1/6*(3*sin(2)^3-10*cos(2)*sin(2)^2+6*cos(2)^2*sin(2)
-12*cos(2)^3)/sin(2)^4*cos(2))/sin(2)*(x-2)^4+(-1/24*(sin(2)-2*cos(2))/s
in(2)-1/60/sin(2)*cos(2)+1/6*(sin(2)^2-cos(2)*sin(2)+2*cos(2)^2)/sin(2)^
3*cos(2)+1/12*(3*sin(2)^3-10*cos(2)*sin(2)^2+6*cos(2)^2*sin(2)-12*cos(2)
^3)/sin(2)^3-1/12*(-10*cos(2)*sin(2)^3+28*cos(2)^2*sin(2)^2+5*sin(2)^4-1
2*cos(2)^3*sin(2)+24*cos(2)^4)/sin(2)^5*cos(2))/sin(2)*(x-2)^5+(1/360-1/
120*(sin(2)-2*cos(2))/sin(2)^2*cos(2)-1/24*(sin(2)^2-cos(2)*sin(2)+2*cos
(2)^2)/sin(2)^2+1/36*(3*sin(2)^3-10*cos(2)*sin(2)^2+6*cos(2)^2*sin(2)-12
*cos(2)^3)/sin(2)^4*cos(2)+1/24*(-10*cos(2)*sin(2)^3+28*cos(2)^2*sin(2)^
2+5*sin(2)^4-12*cos(2)^3*sin(2)+24*cos(2)^4)/sin(2)^4-1/120*(25*sin(2)^5
-122*cos(2)*sin(2)^4+140*cos(2)^2*sin(2)^3-360*cos(2)^3*sin(2)^2+120*cos
(2)^4*sin(2)-240*cos(2)^5)/sin(2)^6*cos(2))/sin(2)*(x-2)^6+(1/240*(25*si
n(2)^5-122*cos(2)*sin(2)^4+140*cos(2)^2*sin(2)^3-360*cos(2)^3*sin(2)^2+1
20*cos(2)^4*sin(2)-240*cos(2)^5)/sin(2)^5+1/72*(-10*cos(2)*sin(2)^3+28*c
os(2)^2*sin(2)^2+5*sin(2)^4-12*cos(2)^3*sin(2)+24*cos(2)^4)/sin(2)^5*cos
(2)-1/144*(3*sin(2)^3-10*cos(2)*sin(2)^2+6*cos(2)^2*sin(2)-12*cos(2)^3)/
sin(2)^3+1/720*(sin(2)-2*cos(2))/sin(2)+1/2520/sin(2)*cos(2)-1/120*(sin(
2)^2-cos(2)*sin(2)+2*cos(2)^2)/sin(2)^3*cos(2)-1/360*(61*sin(2)^6-183*co
s(2)*sin(2)^5+662*cos(2)^2*sin(2)^4-540*cos(2)^3*sin(2)^3+1320*cos(2)^4*
sin(2)^2-360*cos(2)^5*sin(2)+720*cos(2)^6)/sin(2)^7*cos(2))/sin(2)*(x-2)
^7+(1/720*(25*sin(2)^5-122*cos(2)*sin(2)^4+140*cos(2)^2*sin(2)^3-360*cos
(2)^3*sin(2)^2+120*cos(2)^4*sin(2)-240*cos(2)^5)/sin(2)^6*cos(2)-1/288*(
-10*cos(2)*sin(2)^3+28*cos(2)^2*sin(2)^2+5*sin(2)^4-12*cos(2)^3*sin(2)+2
4*cos(2)^4)/sin(2)^4+1/5040*(sin(2)-2*cos(2))/sin(2)^2*cos(2)-1/20160+1/
720*(sin(2)^2-cos(2)*sin(2)+2*cos(2)^2)/sin(2)^2-1/720*(3*sin(2)^3-10*co
s(2)*sin(2)^2+6*cos(2)^2*sin(2)-12*cos(2)^3)/sin(2)^4*cos(2)+1/720*(61*s
in(2)^6-183*cos(2)*sin(2)^5+662*cos(2)^2*sin(2)^4-540*cos(2)^3*sin(2)^3+
1320*cos(2)^4*sin(2)^2-360*cos(2)^5*sin(2)+720*cos(2)^6)/sin(2)^6-1/5040
*(427*sin(2)^7-10080*cos(2)^7+4634*sin(2)^5*cos(2)^2-14532*sin(2)^4*cos(
2)^3+9240*sin(2)^3*cos(2)^4-21840*sin(2)^2*cos(2)^5+5040*cos(2)^6*sin(2)
-2770*cos(2)*sin(2)^6)/sin(2)^8*cos(2))/sin(2)*(x-2)^8+(-1/2880*(25*sin(
2)^5-122*cos(2)*sin(2)^4+140*cos(2)^2*sin(2)^3-360*cos(2)^3*sin(2)^2+120
*cos(2)^4*sin(2)-240*cos(2)^5)/sin(2)^5-1/40320*(sin(2)-2*cos(2))/sin(2)
-1/181440/sin(2)*cos(2)+1/5040*(sin(2)^2-cos(2)*sin(2)+2*cos(2)^2)/sin(2
)^3*cos(2)+1/4320*(3*sin(2)^3-10*cos(2)*sin(2)^2+6*cos(2)^2*sin(2)-12*co
s(2)^3)/sin(2)^3-1/1440*(-10*cos(2)*sin(2)^3+28*cos(2)^2*sin(2)^2+5*sin(
2)^4-12*cos(2)^3*sin(2)+24*cos(2)^4)/sin(2)^5*cos(2)+1/2160*(61*sin(2)^6
-183*cos(2)*sin(2)^5+662*cos(2)^2*sin(2)^4-540*cos(2)^3*sin(2)^3+1320*co
s(2)^4*sin(2)^2-360*cos(2)^5*sin(2)+720*cos(2)^6)/sin(2)^7*cos(2)+1/1008
```

```
0*(427*sin(2)^7-10080*cos(2)^7+4634*sin(2)^5*cos(2)^2-14532*sin(2)^4*cos
(2)^3+9240*sin(2)^3*cos(2)^4-21840*sin(2)^2*cos(2)^5+5040*cos(2)^6*sin(2
)-2770*cos(2)*sin(2)^6)/sin(2)^7-1/20160*(1385*sin(2)^8-5540*cos(2)*sin(
2)^7+24568*cos(2)^2*sin(2)^6-29064*cos(2)^3*sin(2)^5+83664*cos(2)^4*sin(
2)^4-43680*cos(2)^5*sin(2)^3+100800*cos(2)^6*sin(2)^2-20160*cos(2)^7*sin
(2)+40320*cos(2)^8)/sin(2)^9*cos(2))/sin(2)*(x-2)^9
```

(2) 多元函数的完全泰勒展开

① f=xyz　　　　(2,2,4)处 7 阶展开式

```
maple('mtaylor(x*y*z,[x=2,y=2,z=4],7)')
ans =
-32+4*z+8*y+8*x+4*(x-2)*(y-2)+2*(z-4)*(x-2)+2*(y-2)*(z-4)+(x-2)*(y-2
)*(z-4)
```

② f=xy+ytgz　　　　(3,4,5) 处 4 阶展开式

```
maple('mtaylor(x*y+y*tan(z),[x=3,y=4,z=5],4)')
ans =
4*tan(5)+4*x+(3+tan(5))*(y-4)+(4+4*tan(5)^2)*(z-5)+(x-3)*(y-4)+4*tan
(5)*(1+tan(5)^2)*(z-5)^2+(y-4)*(1+tan(5)^2)*(z-5)+(y-4)*tan(5)*(1+tan(5)
^2)*(z-5)^2+(16/3*tan(5)^2+4*tan(5)^4+4/3)*(z-5)^3
```

c. f=sin(x*y*z)/x+y*cos(z*x)　　　(10,20,30)处 5 阶展开式

```
maple('mtaylor(sin(x*y*z)/x+y*cos(z*x),[x=10,y=20,z=30],5)')
ans =
(-45000*cos(6000)+90000000*sin(6000))*(z-30)*(y-20)^3+1/10*sin(6000)
+20*cos(300)+(539999970*sin(6000)-360000*cos(6000))*(z-30)*(x-10)*(y-20)
^2+(-540000*cos(6000)+1080000000*sin(6000)+4500*sin(300)-30*cos(300))*(z
-30)*(y-20)*(x-10)^2+(-3600000*cos(6000)-1200*sin(6000)-300*cos(300)-sin
(300))*(z-30)*(x-10)*(y-20)+(359999980*sin(6000)-240000*cos(6000)+1500*s
in(300)-10*cos(300))*(z-30)^2*(x-10)*(y-20)+(-2000*sin(6000)-1000*cos(30
0))*(z-30)^2+(30*cos(6000)+cos(300))*(y-20)-450000*cos(6000)*(y-20)^3+(2
5000/3*cos(300)+20000000/3*sin(6000))*(z-30)^4+(20*cos(6000)-200*sin(300
))*(z-30)-4500*sin(6000)*(y-20)^2+33750000*sin(6000)*(y-20)^4+(-1/100*si
n(6000)-600*sin(300)+60*cos(6000))*(x-10)+(90000*sin(300)-17999997/5*cos
(6000)+17999999/10000*sin(6000))*(x-10)^3+(-12000*sin(6000)-20*sin(300)-
6000*cos(300))*(z-30)*(x-10)+(-900000*cos(6000)-300*sin(6000))*(z-30)*(y
-20)^2+(80000000*sin(6000)+1000*sin(300)-80000/3*cos(6000)+100000*cos(30
0))*(z-30)^3*(x-10)+(90000*sin(300)-3600000*cos(6000)-600*cos(300))*(z-3
0)*(x-10)^2+(53999982000001/100000*sin(6000)+17999997/50*cos(6000)+67500
0*cos(300))*(x-10)^4+(-400000/3*cos(6000)+10000/3*sin(300))*(z-30)^3+(-1
7999999/1000*sin(6000)-9000*cos(300)-6*cos(6000))*(x-10)^2+(6000*sin(300
)-120000*cos(6000)+449990*cos(300)+360000000*sin(6000))*(z-30)^2*(x-10)^
2+(-50*cos(300)-600000*cos(6000)-200*sin(6000))*(z-30)^2*(y-20)+(7200000
00*sin(6000)+900000*cos(300)+9000*sin(300))*(z-30)*(x-10)^3+(-6000*sin(6
000)+cos(6000)-10*sin(300))*(z-30)*(y-20)+(40000000*sin(6000)+500/3*sin(
300)-20000*cos(6000))*(z-30)^3*(y-20)+(-60000*cos(6000)+89999995*sin(600
0))*(z-30)^2*(y-20)^2+(-1200000*cos(6000)+30000*sin(300)-200*sin(6000)-2
```

```
00*cos(300))*(z-30)^2*(x-10)+(-18000*sin(6000)-30*sin(300))*(x-10)*(y-20
)+(-5400000*cos(6000)-450*cos(300))*(y-20)*(x-10)^2+(-2700000*cos(6000)-
450*sin(6000))*(x-10)*(y-20)^2+(4500*sin(300)+1080000000*sin(6000))*(x-1
0)^3*(y-20)+(-270000*cos(6000)+810000000*sin(6000))*(x-10)^2*(y-20)^2+(2
70000000*sin(6000)-90000*cos(6000))*(x-10)*(y-20)^3
```

(3) 傅里叶级数展开

① **y=x**

```
y=x;
[a0,an,bn]=mfourier(y)
a0 =
0
an =
0
bn =
-2*(-sin(pi*n)+pi*n*cos(pi*n))/n^2/pi
```

② **$y=x^2$**

```
y=x^2;
[a0,an,bn]=mfourier(y)
a0 =
2/3*pi^2
an =
2*(n^2*pi^2*sin(pi*n)-2*sin(pi*n)+2*pi*n*cos(pi*n))/n^3/pi
bn =
0
```

③ **$y=\sin 2x^2$**

```
y=sin(2*x^2);
[a0,an,bn]=mfourier(y)
a0 =
FresnelS(2*pi^(1/2))/pi^(1/2)
an =
(1/4*pi^(1/2)*cos(1/8*n^2)*FresnelS(1/2*(4*pi+n)/pi^(1/2))-1/4*pi^(1
/2)*sin(1/8*n^2)*FresnelC(1/2*(4*pi+n)/pi^(1/2))-1/4*i*pi^(1/2)*cos(1/8*
n^2)*FresnelS(1/2*i*(-4*pi+n)/pi^(1/2))-1/4*i*pi^(1/2)*sin(1/8*n^2)*Fres
nelC(1/2*i*(-4*pi+n)/pi^(1/2))-1/4*pi^(1/2)*cos(1/8*n^2)*FresnelS(1/2*(-
4*pi+n)/pi^(1/2))+1/4*pi^(1/2)*sin(1/8*n^2)*FresnelC(1/2*(-4*pi+n)/pi^(1
/2))+1/4*i*pi^(1/2)*cos(1/8*n^2)*FresnelS(1/2*i*(4*pi+n)/pi^(1/2))+1/4*i
*pi^(1/2)*sin(1/8*n^2)*FresnelC(1/2*i*(4*pi+n)/pi^(1/2)))/pi
bn =
0
```

- **符号方程及方程组的求解**

(1) 求解线性方程组:AX=B

```
A=[45 23 53 65; 21 53 35 21;
```

```
7 23 14 3;  8  3  8  2];
B=[ 3 ; 4 ; 5 ; 6];

X=linsolve(A,B)
X =
[  -4883/25552]
[  -5179/12776]
[    8125/6388]
[ -18275/25552]
```

(2) 求解方程 f= sinx+tgx+1=0

```
f=sym('sin(x)+tan(x)+1=0');

x=solve(f)
x =
[atan((-(-1/2+1/2*2^(1/2)+1/2*(-1+2*2^(1/2))^(1/2))^3-(-1/2+1/2*2^(1/2)+1/2*(-1+2*2^(1/2))^(1/2))^2+1)/(-1/2+1/2*2^(1/2)+1/2*(-1+2*2^(1/2))^(1/2)))]
[atan((-(-1/2+1/2*2^(1/2)-1/2*(-1+2*2^(1/2))^(1/2))^3-(-1/2+1/2*2^(1/2)-1/2*(-1+2*2^(1/2))^(1/2))^2+1)/(-1/2+1/2*2^(1/2)-1/2*(-1+2*2^(1/2))^(1/2)))+pi]
[atan(-(-1/2-1/2*2^(1/2)+1/2*(-1-2*2^(1/2))^(1/2))^3-(-1/2-1/2*2^(1/2)+1/2*(-1-2*2^(1/2))^(1/2))^2+1,-1/2-1/2*2^(1/2)+1/2*(-1-2*2^(1/2))^(1/2))]
[atan(-(-1/2-1/2*2^(1/2)-1/2*(-1-2*2^(1/2))^(1/2))^3-(-1/2-1/2*2^(1/2)-1/2*(-1-2*2^(1/2))^(1/2))^2+1,-1/2-1/2*2^(1/2)-1/2*(-1-2*2^(1/2))^(1/2))]

x=double(x)
x =
  -0.4881
   2.0589
  -2.3562 + 1.1284i
  -2.3562 - 1.1284i
```

(3) 求解方程组:

$$\begin{cases} x+y+z=0 \\ x^2+yz+x=10190 \\ x/y+z/y+y/x+y/z=16327/225 \end{cases}$$

```
f1=sym('x+y+z=0');
f2=sym('x^2+y*z+x= 10190');
f3=sym('x/y+z/y+y/x+y/z= 16327/225');
[x,y,z]=solve(f1,f2,f3);
x=double(x)
x =
  1.0e+002 *
```

```
  -1.0079
   0.9980
   0.0000 - 0.0135i
   0.0000 + 0.0135i

y=double(y)
y =
  1.0e+002 *

   0.9945
  -0.9848
   0.0001 - 1.0028i
   0.0001 + 1.0028i

z=double(z)
z =
  1.0e+002 *

   0.0133
  -0.0132
  -0.0001 + 1.0163i
  -0.0001 - 1.0163i
```

第 7 章

- **多元函数的极限、微分、极值**

(1) 多元函数的极限

① f=xyz/sinz 在(1,2,0)处

```
syms x y z
f=x.*y.*z./sin(z);
limit(limit(limit(f,z,0),y,2),1)
```

② g=xz/sinx+xsiny 在(3,4,7)处

```
syms x y z
g=x*z/sin(x)+sin(y)*x;
limit(limit(limit(g,z,7),y,4),3)
```

(2) 多元函数的求导 附:求梯度的数值求法

① $f=ytg(\sin z)+xye^{x^z})$ 对变量 z 的偏导数;梯度

```
syms x y z
f=y*tan(sin(z))+x*y*exp(x^z);
dfz=diff(f,z)
gradz=jacobian(f,[x,y,z])
```

② $g=[z\sin x\cos y; xy/z; e^{z/xy})$的 Jacobia 矩阵

```
syms x y z
g=[sin(x)*cos(y)*z;x*y/z;exp(z/x*y)];
gradg=jacobian(g,[x,y,z])
```

(3) 显式复合函数微分求导

试求多元复合函数：

```
f1=uvsin(uv/w);          ┌ u=yxz;
f2=e^(v/u-w^2);          ┤ v=zy/x+zx/y+xy/z;
f3=acos(u/v)asinw;       └ w=x2+y2+z2;
```

的关于x, y, z的导数矩阵。

```
u=y*x*z;
v=z*y/x+z*x/y+x*y/z;
w=x^2+y^2+z^2;
f1=u*v*sin(u*v/w);
f2=exp(v/u-w^2);
f3=acos(u/v)*asin(w);
gradf=jacobian([f1,f2,f3],[x y z])
```

(4) 隐函数的求导

① $F=x^3+\sin xy+z\cos z$ 求 dz/dx

② $G=a^2+b^3+c^4+1/abc$ 求 da/dc

```
syms x y z a b c
F=x^3+sin(y*x)+z*cos(z);
G=a^2+b^3+c^4+1/(a*b*c);
dz_dx=diff(F,z)/diff(F,x)
da_dc=diff(G,a)/diff(G,c)
```

(5) 略

(6) 条件极值

① 在曲面 $F=4x^2+2y^2+3z^2-4xy-2yz=0$ 上求 $f=xy\sin z$ 的极值点。

```
syms x y z lambda
f=4*x^2+2*y^2+3*z^2-4*x*y-2*y*z;
F=x*y*sin(z);
v=[x y z];
M=jacobian(F,v);
G=jacobian(f,v);
A=M.'-lambda*G.';
A(4)=f;
[lambda x y z]=solve(A(1),A(2),A(3),A(4))
F
fmin_max=subs(F)
```

② 在抛物线 $y^2=4x$ 上求一点,使其距离(2,7)最近。

```
syms lambda
syms x y real
```

```
f=y^2-4*x;
F=((x-2)^2+(y-7)^2);
v=[x y];
M=jacobian(F,v);
G=jacobian(f,v);
A=M.'-lambda*G.';
A(3)=f;
[lambda x y]=solve(A(1),A(2),A(3))
F
fmin_max=double(subs(F))
```

③ 在椭球 $x^2/3+y^2/4+z^2/5=0$ 内放入一个立方体，求该立方体的最大体积。

```
syms x y z lambda
f=x^2/3+y^2/4+z^2/5-10;
F=(2*x)*(2*y)*(2*z);
v=[x y z];
M=jacobian(F,v);
G=jacobian(f,v);
A=M.'-lambda*G.';
A(4)=f;
[lambda x y z]=solve(A(1),A(2),A(3),A(4))
F
fmin_max=double(subs(F))
```

④ 求两曲面 $z1=x^2+y^2;z2=x+y+10$ 的交线到原点的最近距离。

```
syms x y z lambda1 lambda2
f1=z-x^2+y^2;
f2=z-x+y+10;
F=x^2+y^2+z^2;
v=[x y z];
M=jacobian(F,v);
G1=jacobian(f1,v);
G2=jacobian(f2,v);
A=M.'-lambda1*G1.'-lambda2*G2.';
A(4)=f1;
A(5)=f2;
[lambda1 lambda2 x y z]=solve(A(1),A(2),A(3),A(4),A(5));
F
fmin_max=double(subs(F))
```

- 空间曲线积分

(1) 空间曲线曲面

① 求下列曲线在指定点处的切线及法平面。

a. $x=t-\sin t, y=1-\cos t, z=4\sin(t/4)$ 在$(\pi/2-1,1,2\times 2^{1/2})$处

```
syms t real
x=t-sin(t);
```

```
y=1-cos(t);
z=4*sin(t/4);
f=[x,y,z];
df=diff(f,t);
t0=pi/2;
p0=double([pi/2-1,1,2*2^0.5]');
line=p0+df'*(t-t0)
syms t real
x=t-sin(t);
y=1-cos(t);
z=4*sin(t/4);
f=[x,y,z];
df=diff(f,t);
t0=pi/2;
p0=double([pi/2-1,1,2*2^0.5]');
line=p0+df'*(t-t0)
syms x y z real
t=pi/2;
df=subs(df);
face=df*[x-p0(1),y-p0(2),z-p0(3)]'
```

b. $x^2+y^2+z^2=8;z=x^2+y^2$ 在(2,2,72)处

```
syms x y z
F=x^2+y^2+z^2-8;
G=z-x^2+y^2;
v=[x y z];
gradF=jacobian(F,v);
gradG=jacobian(G,v);
x=2;
y=2;
z=72;
gradF=subs(gradF)
gradG=subs(gradG)
V=cross_product(gradG,gradF)'
line=[2,2,72]'+V*t
```

② 求下列曲面在指定点处的法向量及切平面。

a. $x^2+y^2+z^3=3$ 在(1,1,1)处

```
syms x y z
F=x^2+y^2+z^3-3;
n=jacobian(F,[x y z])
x=1;
y=1;
z=1;
n=subs(n);
syms x y z
```

```
face=n*[x-1;y-1;z-1]
```

b. $z=x^2-y^2$　　　　在(4,1,15)处

```
syms x y z
F=z-(x^2-y^2);
n=jacobian(F,[x y z])
x=4;
y=1;
z=15;
n=subs(n);
syms x y z
face=n*[x-4;y-1;z-15]
```

(2) 第二型曲线积分

试求向量函数 $B=[x^2/y;\sin(yz);z]$在有向曲线 $x=t^2, y=\sin t+2, z=t$；$t\in[2,8]$上的积分。

```
syms x y z t
x=t^2;
y=sin(t)+2;
z=t;
B=[x^2/y;sin(y*z);z];
F=B'*diff([x y z],t)'
int(F,t,2,8)
```

- LALACE FOURIER AND Z 的变换及逆变换

(1) 完成函数 $f=1/xe^{-|x|}+x$ 的 Fourier 变换及其结果的逆变换。

(2) 完成函数 $f=\sin(xt+2t)+e^x+x$ 的 Laplace 变换及其结果的逆变换。

(3) 完成函数 $f=1/xe^{-|x|})+x$ 的 Z 变换及其结果的逆变换。

```
syms x t y
f=1/x*exp(-abs(x))+x;
F=fourier(f)
ff=ifourier(F)
f=sin(x*t+2*t)+exp(x)+x;
L=laplace(f)
lf=ilaplace(L)
f=sin(x*t+x)+exp(x/t);
Z=ztrans(f)
zf=iztrans(Z)
```

- 常微分方程及方程组的求解

(1) 求解下列微分方程

① $y'=(x+y)(x-y)$

② $xy'=y\log(y/x)$　　　　$y(10)=1$

③ $y'=-x\sin x/\cos y$　　　　$y(2)=1$

```
syms x t y
```

```
diff_equ='Dy=(x+y)*(x-y)';
y1=dsolve(diff_equ,'x')
diff_equ='x*Dy=y*log(y/x)';
y2=dsolve(diff_equ,'y(10)=1','x')
diff_equ='Dy=-x*sin(x)/cos(y)';
y3=dsolve(diff_equ,'y(2)=1','x')
```

(2) 求解下列微分方程组

① f'=f+3g,g'=f+4

```
[f,g]=dsolve('Df=f+g*3,Dg=f+4')
```

② X'=AX 其中

```
A=[2 44 8 3;3 4 2 9;9 23 8 43;92 4 1 4]
A=[2 44 8 3;3 4 2 9;9 23 8 43;92 4 1 4];
X=dsolves(A)
```

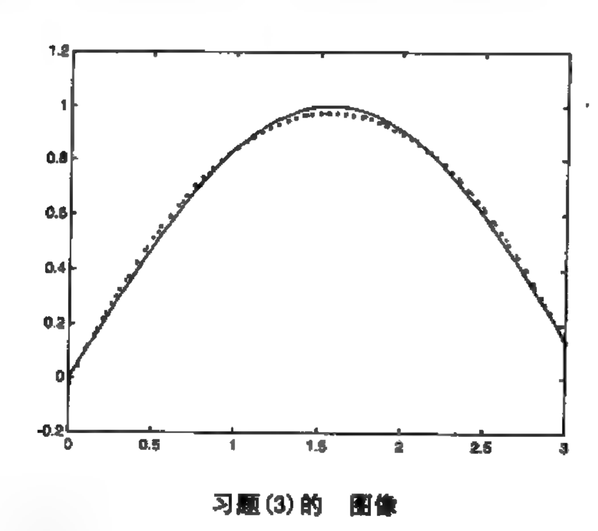
第8章

(1) 3.0325

(2) 最小二乘法：1.49383916990920　　3.03245784695202

多项式拟合法：1.49383916990921　　3.03245784695201

(3)



习题(3)的　图像

(4) 对其中七个点进行拟合：

x=0:0.5:3;

y=sin(x);

(5) 拉格朗日法：

-0.0013　0.0123　-0.0065　-0.1606　-0.0029　1.0005　0

牛顿法：-0.0013　0.0123　-0.0065　-0.1606　-0.0029　1.0005　0

牛顿一格雷高里法：-0.0013　0.0123　-0.0065　-0.1606　-0.0029　1.0005　0

```
5.f =
```

```
ff0
ff1
ff2
ff3
ff4
mini_product(f,x,y)
ans =
    0.0001
    0.9856
    0.0524
   -0.2334
    0.0372
polyfit(x,y,4)
ans =
    0.0372   -0.2334    0.0524    0.9856    0.0001
```

(6) 拉格朗日法：1.8147

牛顿法：1.8147

```
interp1(x,y,2,'nearest')
ans =
     2
interp1(x,y,2,'linear')
ans =
    1.8833
interp1(x,y,2,'spline')
ans =
    1.8461
```

(7) p=polyfit(x,y,5);

```
pp=polyder(p);
polyval(pp,2)
ans =
   -1.7819
  function y=abc(xx)
x=[0.1 0.8 1.3 1.9 2.5 3.1];
y=[1.2 1.6 2.7 2.0 1.3 0.5];
p=polyfit(x,y,5);
y=polyval(p,xx);

centdiff('abc',2,0.01,1e-4)
ans =
   -1.7819
```

(8) 低阶法：

```
quad('abc',0.1 , 3 , 1e-6 )
ans =
```

```
    4.6888
```

高斯十点法：

```
gussled('abc',0.1 , 3 )
ans =
    4.8839
```

似抛物线公式：

```
guass('abc',0.1 , 3,50 )
ans =
    4.6888
```

复合辛普生法：

```
comsimpson('abc',0.1 , 3,50 )
ans =
    4.6888
```

第 9 章

(1) 试用 subplot 命令在同一图形输出窗口中绘出如下 3 个函数的图形。

① $y=x^2$　[-2,2]

② $y=x^{1/2}$　[0,4]

③ $y=x^{-1/2}$　[0,4]

```
x1=-2:0.1:2;
y1=x1.^2;
x2=0:0.1:4;
y2=x2.^0.5;
y3=x2.^(-0.5);
subplot(1,3,1)
plot(x1,y1)
subplot(1,3,2)
plot(x2,y2)
subplot(1,3,3)
plot(x2,y3)
```

(2) 试绘出颜色为洋红色，数据点用五角星标出的函数 $y=x.*e^{\sin x}$ 在[0,5]上的虚线图。

```
x=0:0.1:5;
y=x.*exp(sin(x));
plot(x,y,'mp--')
```

(3) 绘出函数 $f=x^2+e^y|x|$ 的在[-2,2], [-2,2]上的三维瀑布图。

```
x=-2:0.1:2;
[x,y]=meshgrid(x);
f=x.^2+exp(y).*abs(x);
waterfall(x,y,f)
```

(4) 绘出四维函数 f=xsin(y/z)的四维表现图。

```
x=-2:0.1:2;
y=0:0.1:4;
z=2:0.1:6;
[X,Y,Z]=meshgrid(x,y,z);
W=X.*sin(Y./Z);
slice(X,Y,Z,W,[0 2],[4],[2 4])
```

(5) 绘出矢量场 z=x.*y+e^{x} 的空间矢量场图，并对图形进行标注。

```
x=-2:0.2:2;
[x,y]=meshgrid(x);
z=x.*y+exp(x);
surf(x,y,z)
hold on
[U,V,W]=surfnorm(x,y,z);
quiver3(x,y,z,U,V,W);
view([-3,-0,15])
```

(6) 试用线性纯铜色重绘题 3 中函数的表面图。

```
R=1;
G=0.62;
B=0.4;
colormap([R,G,B])
x=-2:0.4:2;
[x,y]=meshgrid(x);
z=x.*y+exp(x);
surf(x,y,z)
```

(7) 完成函数 $z=x^2+y^2\sin x$ 的动画表达。

```
x=-3:0.1:0;
[x,y]=meshgrid(x);
z=x^2.*sin(y)+y^2*sin(x);
mesh(z);
M=moviein(30);
axis manual
set(gca,'nextplot','replacechildren');
for j = 1:30
    mesh(cos(4*pi*j/30)*z,z)
  M(:,j)=getframe;
end
movie(M,25)
```

(8) 绘出函数 $x=\sin t, y=t^2+e^t$ 的彗星效果图。

```
x=0:0.1:50;
y=sin(x);
z=x.^2+exp(y);
```

```
comet3(x,y,z)
```

第11章

(1)
```
Include<stdlib.h>
Include<stdio.h>
Include<string.h>
Include "engine.h"
Void main()
{ mxArray  *mp,*mp2;
double *Ar,*Br;
MATFile *fp1,*fp2;
Int i,j,m,n;
fp1=matOpen("A.mat","r");
if (fp1 == NULL) {
    printf("Error creating file A.mat \n");
    printf("(do you have write permission in this directory?)\n");
    return(1);
  }
fp2=matOpen("B.mat","w");
if (fp2 == NULL) {
    printf("Error creating file B.mat \n");
    printf("(do you have write permission in this directory?)\n");
    return(1);
  }
mp=matGetArray(fp1,"B");
Ar=mxGetPr(mp);
   m=mxGetM(Ar);
   n=mxGetN(Ar);
       mp2= mxCreateDoubleMatrix (m,n,mxREAL);
       Br=mxGetPr(mp2);
   mxSetName(mp2,"B");
   for(i=1;i<=m;i++)
      for(j=1;j<=n;j++)
      Br[i-1+n*(j-1)]=2*Ar[i-1+n*(j-1)];
   MatPutArray(fp2,mp2);
   MatDeleteArray(fp1,"A");
       mxDestroyArray(mp);
      mxDestroyArray(mp2);
 if (matClose(fp1) != 0) {
  printf("Error closing file %s\n","A.mat");
  return(1);
}
 if (matClose(fp2) != 0) {
  printf("Error closing file %s\n","B.mat");
  return(1);
}
```

```
Exit(0);
}
```

(2)

```
      program plotm
      integer engOpen,engClose,engEvalString,mxCreateFull
      integer engPutMatrix,mp,ep,stat,i,j
      double precision A(5,5)
c 生成矩阵
      do 50 i=1,5
        A(i,1)=i
50    continue
      do 100 i=1,5
        do 110 j=1,5
          A(i,j)=power(A(i,j),j)
110     continue
100   continue
c 打开一个MatLab引擎
      ep=engOpen('')
c 将矩阵命名A,并以MatLab格式保存
      mp=mxCreateFull(5,5,0)
      call mxSetName(mp,"A")
      call mxCopyReal8ToPtr(A,mxGetPr(mp),25)
c 将矩阵保存在MatLab工作空间中
      stat=engPutMatrix(ep,mp)
c 将矩阵A所代表的图形画出
      stat=engEvalString(ep,"mesh(A);")
      stat=engEvalString(ep,"pause(10);")
c 将图形保存在一个文件中
      stat=engEvalString(ep,"print Images/ex154.eps -deps")
c 关闭MatLab引擎
      stat=engClose(ep)
      stop
      end
```

(3) 由于有关高斯消元法的介绍很多，可以通过其他高级语言的高斯消元法将它改造成为C或FORTRAN语言的MEX函数，在此将不给出程序的源代码。

(4)

```
#include <math.h>
#include "mex.h"
/* 定义输入参数 */
#define Y_IN       prhs[0]
/* 定义输出参数 */
#define YP_OUT  plhs[0]
void mexFunction(int nlhs,mxArray *plhs[],int nrhs, const mxArray
*prhs[])
```

```
{
  unsigned int  m,n;
  MxArray *ap,*oap[1],*iap[1];
  Int no,ni;
  /* 检查参数个数的合法性 */
  if (nrhs != 1) {
    mexErrMsgTxt("requires one input arguments.");
  } else if (nlhs > 1) {
    mexErrMsgTxt("requires one output argument.");
  }
  /* 检查参数维数的合法性 */
  m = mxGetM(Y_IN);
  n = mxGetN(Y_IN);
  if (!mxIsNumeric(Y_IN) ||.mxIsComplex(Y_IN) ||
      mxIsSparse(Y_IN)  || !mxIsDouble(Y_IN) ||
      (m!=n) {
    mexErrMsgTxt("the argument should be a square.");
  }
  /* 将输入参数指定指针变量 */
  iap[0] = mxGetPr(Y_IN);
  /* 对数据进行计算 */
no=1;ni=1;
MexCallMatLab(no,oap,ni,iap,"inv");
ap=mxCreateDoubleMatrix(m,n,mxREAL);
Memcpy(mxGetPr(ap),oap[0],m*n*sizeof(double));
Y_OUT=ap;
  return;
}
```

(5) 请参照第 8 章中给出的数值积分和数值微商的 MatLab 语言的函数，以及本章中介绍的编写 MEX 函数的具体步骤和方法，自行将这些函数改造成 C 或 FORTRAN 语言的 MEX 函数。函数的算法与给出的函数基本相同。

